GOAL PROGRAMMING ALGORITHMS FOR NETWORK FLOW PROBLEMS

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY

by
A. SRINIVAS

to the

INDUSTRIAL AND MANAGEMENT ENGINEERING PROGRAMME
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

APRIL, 1985

2/4/85

ii

CERTIFICATE

This is to certify that the present work on "Goal Programming Algorithms for Network Flow Problems," by A. Srinivas has been carried out under my supervision and has not been submitted elsewhere for the award of a degree.

RKAhuja

(R. K. Ahuja)
Lecturer
Industrial and Management Engg.Program
Indian Institute of Technology
Kanpur 208 016, INDIA

April, 1985

ACKNOWLEDGEMENTS

I am grateful to my thesis supervisor, Dr. R.K. Ahuja for his valuable suggestions and excellent guidance throughout the span of this work.

I am also thankful to my friend Mr. V.V.S.N. Murthy for his help during the course of this work.

Lastly, I thank Mr. J.K. Misra for his excellent typing of this manuscript, and Mr. Buddhi Ram Kandiyal for his neat cyclostyling work.

A. Srinivas

CONTENTS

ABSTRACT

In this dissertation, we consider the weighted goal programming and interval goal programming problems in the network context. Special structure embedded in their problems is used to develop computationally efficient algorithms. In weighted goal network flow [WGNF] problem, the decision maker specifies aspiration levels for each of the objectives and weighting factors for each of the deviations from the aspiration levels. On the other hand, in interval goal network flow [IGNF] problem, the decision maker provides a range of aspiration levels for each objective and weighting factors for each of the deviations from the specified range of aspiration levels. Both the algorithms developed trace a path in the feasible region and obtain the solutions which minimize the sum of weighted deviations. The WGNF and the IGNF algorithms are based on the parametric approach and fully exploit the special structure of the minimum cost flow problem in order to perform all the computations on the network itself. Computer programs were written for both the algorithms, and tested on randomly generated network problems. Results of their investigations are presented.

CHAPTER I

INTRODUCTION

1.1 Introduction:

Operations Research is a scientific approach to solve complex problems arising in the management of large systems of men, machines, materials and money. Operations Research is a decision science which helps management to make better decisions. Network model is a branch of study in Operations Research. Due to the wide applicability of network models in real world, it is considered as an important branch of study in Operations Research. In the present day, we find that complex intriguing problems arising in production-distribution systems, military logistics systems, urban traffic systems, railway systems, communication systems, pipeline network systems, facilities location systems, file merge systems, electrical networks etc. can be tackled by constructing an appropriate network model. Furthermore, network geometry (or relationships) can be easily displayed in two-dimensional drawings, greatly simplifying the communication problem between the analyst and the client for whom the model is designed. In the recent years, network flow problems have received special attention due to significant

advances in implementation technology and solution techniques, thereby increasing the applicability of the network models substantially.

In this dissertation, we consider the bicriteria minimum cost flow problems and propose two algorithms i.e. weighted goal network flow (WGNF) and interval goal network flow (IGNF) algorithms for obtaining optimum solutions. These models are useful when two conflicting objectives are to be simultaneously considered, i.e. one is interested in minimizing the total cost as well as total time.

One application of the bicriteria minimum cost flow problem arises in a distribution system. Suppose V units of perishable items (i.e. fruits) are to be sent from source to sink along the road network. The two objectives to be considered simultaneously are total cost and total time. We would like to transport the items in such a way that they are not perished before they reach the sink and at the same time the total cost of transportation should be minimum.

Another application of the bicriteria minimum cost flow problem arises in communication networks. Suppose V units of messages/sec. are required to be transmitted from source to sink along a communication network. The decision maker would like to consider two objectives simultaneously. One of the objectives may be to minimize total cost of sending messages,

and the other being the minimization of total amplification cost.

Another important application of bicriteria network flow problem is in the area of flow through a pipeline network. The decision maker would like to minimize the total operating cost for sending the flow as well as minimize the total time to transmit flow from source to sink.

In this work, we develop specializations of goal programming i.e. weighted goal and interval goal network flow algorithms based on parametric approach. The special structure of minimum cost flow problem is exploited to perform all the computations on the network itself. Efficient tree data structures are used to further enhance the efficiency of these computations. Computational investigations with the above approaches are found to be very encouraging and presented in sufficient detail.

1.2 Outline of the Thesis:

In this section, we give a brief outline of this thesis.

In Chapter II, we present some of the preliminary knowledge required to understand the work done in the subsequent chapters. We first review the literature related to the bicriteria minimum cost flow problem. The graph theory notations adopted in this work are also described. Finally, an

overview of the bicriteria network flow problem and the goal programming are briefly outlined.

Chapter III embodies the main work done on WGNF problem and IGNF problem. They can be formulated as shown below:

Weighted Goal Network Flow Problem:

$$\text{Min.} \quad Z = w_1\alpha_1 + w_2\alpha_2 + r_1\beta_1 + r_2\beta_2$$

s.t.

$$\sum_{(j,i)\varepsilon I(i)} x_{ji} - \sum_{(i,j)\varepsilon O(i)} x_{ij} = \begin{cases} -V, & \text{if } i=1 \\ V, & \text{if } i=n, \quad \forall\, i\varepsilon N \\ 0, & \text{Otherwi} \end{cases}$$

$$0 < x_{ij} \leq b_{ij}, \quad \forall \quad (i,j)\ \varepsilon\ A$$

$$\sum_{(i,j)\varepsilon A} c_{ij}\, x_{ij} \quad \beta_1 - \alpha_1 = C_1$$

$$\sum_{(i,j)\varepsilon A} d_{ij}\, x_{ij} + \beta_2 - \alpha_2 = D_1$$

Interval Goal Network Flow Problem:

$$\text{Min.} \quad Z = w_3\alpha_3 + w_4\alpha_4 + r_1\beta_1 + r_2\beta_2$$

s.t.

$$\sum_{(j,i)\varepsilon I(i)} x_{ji} - \sum_{(i,j)\varepsilon O(i)} x_{ij} = \begin{cases} -V, & \text{if } i = 1 \\ V, & \text{if } i = n, \ \forall\ i\varepsilon N \\ 0, & \text{Otherwise} \end{cases}$$

$$0 \leq x_{ij} \leq b_{ij}, \quad \forall \quad (i,j)\ \varepsilon\ A$$

$$\sum_{(i,j)\varepsilon A} c_{ij}\, x_{ij} + \beta_1 - \alpha_1 = C_1$$

$$\sum_{(i,j)\varepsilon A} d_{ij}\, x_{ij} + \beta_2 - \alpha_2 = D_1$$

$$\sum_{(i,j)\varepsilon A} c_{ij} \, x_{ij} + \beta_3 - \alpha_3 = C_2$$

$$\sum_{(i,j)\varepsilon A} d_{ij} \, x_{ij} + \beta_4 - \alpha_4 = D_2$$

where,

x_{ij} is the amount of flow on arc (i,j).

c_{ij} is the cost of unit flow on arc (i,j).

d_{ij} is the budget required for unit flow on arc (i,j)

b_{ij} is the capacity of arc (i,j).

V is the net flow of commodity from source to sink n.

C_i is the aspiration level for cost objective function.

D_i is the aspiration level for budget objective function.

α_i is the positive deviation.

β_i is the negative deviation.

We suggest two exact algorithms to solve bicriteria network flow problem. They are weighted goal network flow (WGNF) and interval goal network flow (IGNF) algorithms. Both these algorithms are based on the parametric algorithm for constrained minimum cost flow problem [1].

The WGNF algorithm searches for an optimum solution which satisfies the aspiration levels specified for each objective by the decision maker. Whereas IGNF algorithm searches for an optima that lies in the specified range of aspiration levels for each of the objectives.

The computational investigations of both the algorithms are given at the end of Chapter III. Computer programs are written for both the algorithms and the computational performances are ascertained by solving different sized problems. Efficient data structures are implemented to represent the basis tree which requires comparatively less storage. Data structures based on augumented threaded index method are used and results of the computational investigations are presented.

CHAPTER II

PRELIMINARIES

2.1 Introduction:

In this chapter, we review some of the relevent concepts of the bicriteria minimum cost flow problem and goal programming. The basic reason behind this is to prepare a sufficient background which will help in understanding the algorithms developed in the subsequent chapters.

This chapter is divided into five sections, The graph theory notations adopted in this work are presented in Sec. 2. The literature related to the bicriteria minimum cost flow problem is surveyed in Sec. 3. A brief review of various methods of solving bicriteria minimum cost flow problem and a parametric algorithm for solving constrained minimum cost flow problem [1] are presented in Sec. 4. In Sec. 5 a brief description about goal programming and its importance to management in decision making to real world situations, and various goal programming techniques are discussed.

2.2 Graph Theory Notations:

Some notations and well-known concepts of graph theory that are being used throughout the thesis are mentioned below.

A *directed graph* $G = (N, A)$, consists of a finite set N of elements, called *nodes*, and a set A of ordered pairs of nodes called *arcs*. A *directed network* is a directed graph in which numerical values are attached to the nodes and arcs of the graph. Let $n = |N|$ and $m = |A|$. The two specified nodes 1 and n are called the *source* and the *sink* respectively.

An arc (i,j) has two end points, i and j, and it is said to be *incident from* node i and *incident to* node j. Let $I(i)$ and $O(i)$ denote, respectively, the sets of arcs incident to and incident from node i. The degree of a node i is the number of arcs incident to or incident from that node.

A *path* in $G = (N, A)$ is a sequence $i_1, i_2, \ldots, i_r$ of distinct nodes of N such that either $(i_k, i_{k+1}) \varepsilon A$ or $(i_{k+1}, i_k) \varepsilon A$ for each $k = 1,\ldots, r-1$. A *directed path* is defined similarily, except that $(i_k, i_{k+1}) \varepsilon A$ for each $k = 1,\ldots, r-1$. A cycle is a path together with an arc (i_r, i_1) or (i_i, i_r). A *directed cycle* is a directed path together with the arc (i_r, i_i).

A graph $G = (N', A')$ is a subgraph of $G = (N, A)$ if $N' \subseteq N$ and $A' \subseteq A$. A graph $G = (N', A')$ is a spanning subgraph of $G = (N, A)$ if $N' = N$ and $A' \subseteq A$.

Two nodes i and j are said to be *connected* if there is atleast one path between them. A graph is said to be

<u>connected</u> if all pairs of nodes are connected, otherwise it is called <u>disconnected</u>. A set $Q \subseteq A$ such that the graph $G = (N, A-Q)$ disconnected and no subset of Q has this property, is called a cocycle of G. A cocycle is a cutset if it disconnects source and sink.

A graph is <u>acyclic</u> if it does not contain any cycle. A <u>tree</u> is a connected acyclic graph. A <u>subtree</u> of a tree T is a subgraph of T as well as a tree. A tree T is said to be a <u>spanning tree</u> of G if T is a spanning subgraph of G. Arcs belonging to a spanning tree T are called <u>tree-arcs</u> and arcs not belonging to T are called <u>nontree-arcs</u>. A spanning tree of $G = (N, A)$ has exactly $(n-1)$ tree-arcs.

2.3 <u>Literature Review</u>:

One of the methods of solving the bicriteria minimum cost flow problem is to consider one of the objectives as a constraint and solve the resultant constrained minimum cost flow [CMCF] problem. We will first present a brief review of the constrained flow problems.

Hultz and Klingman [9] have suggested a partitioning method in conjunction with the simplex method for solving constrained generalized network flow problems. Takashi Kobayashi [12] proposed a primal-dual method for solving CMCF problem. He has associated two dimensional distance for each arc. The first element is related to the cost and the second

one to the coefficient of the additional constraint. This method is suitable in cases when degeneracies often occur.

A parametric algorithm for solving CMCF problem is developed by Ahuja, Batra and Gupta [1]. The algorithm uses the concepts from parametric linear programming and fully exploits the topological structure embedded in the problem. The algorithm can also be used to generate all the efficient points of the bicriteria minimum cost flow problem.

Chen and Saigal [4] have suggested a primal algorithm for solving a capacitated network flow problems with additional linear constraints.

Klingman and Mote [10] have reviewed the fundamental theoretical results for the general multi-criteria linear programming and the relevent exploitable characteristics of the network basis. Two algorithms are then developed for solving the multicriteria network flow problems efficiently. One approach determines the set of all non-dominated solutions to the problem. The other approach is a network variant of the surrogate criterion linear programming approach. Ignizio [7] has suggested a straight forward weighted integer goal programming for generalized networks models for integer programming problems. His method is simplex and robust.

Practically no study was made in the field of application of goal programming techniques to network flow problems. This has motivated us to propose two algorithms for WGNF and IGNF problems.

2.4 Bicriteria Linear Programming:

Bicriteria linear programming deals with the optimization of two objective functions (multicriteria deals with two or more than two objective functions) simultaneously. A decision situation is generally characterised by multiple objectives. Some of these objectives may be complementary, while others may be conflicting in nature.

The bicriteria minimum cost flow problem differs from the classical minimum cost flow problem only in the expression of their objective functions. The bicriteria minimum cost flow problem is shown as below:

$$\text{Minimize} \quad \left\{ \begin{array}{l} \sum_{(i,j)\varepsilon A} c_{ij}\, x_{ij} \\ \sum_{(i,j)\varepsilon A} d_{ij}\, x_{ij} \end{array} \right\}$$

subject to

$$\sum_{(j,i)\varepsilon I(i)} x_{ji} - \sum_{(i,j)\varepsilon O(i)} x_{ij} = \begin{cases} -V, & \text{if } i=1 \\ V, & \text{if } i=n, \quad \forall\, i \varepsilon N \\ 0, & \text{otherwise} \end{cases}$$

$$0 \leq x_{ij} \leq b_{ij}, \qquad \forall\, (i,j)\ \varepsilon\ A$$

Generally, it is observed that these two objectives are conflicting in nature and hence there is no optimal solution in the normal sense for the above problem. The decision maker has to choose solutions, possibly not the best for both the criteria. A special set of solutions, the <u>non-dominated</u> or <u>efficient solutions</u> can be defined to overcome this problem.

An <u>efficient solution</u> is one in which one objective cannot be reduced without a simultaneous increment in the other objective. That is, X^* is an efficient solution to the bicriterion minimum cost flow problem if there does not exist any $X^o \in S$, the set of all possible solutions such that

$$Z_1(X^o) \leq Z_1(X^*) \quad \text{and}$$
$$Z_2(X^o) \leq Z_2(X^*)$$

with atleast one strict inequality.

A number of techniques as mentioned below are available for generating efficient or non-inferior solutions for the bicriteria problem formulated above.

(i) The weighting method

(ii) The constraint method

(iii) The non-inferior set estimation method, and

(iv) The multi-objective simplex algorithm.

These techniques are discussed in detail by Ambrose Goicoechea [3].

There is one disadvantage with non-dominated solutions technique. It becomes difficult for the decision maker to make his final choice from a set of non-dominated solutions. Another technique known as goal programming, allows the decision maker to specify a target for each of the objective functions. It obtains a preferred solution which is defined as the one that minimizes the sum of the deviations from the prescribed set of target values. A brief description of the goal programming is presented in Sec. 2.5.

An Overview of Constrained Minimum Cost Flow Problems

We now briefly outline the parametric algorithm for the constrained minimum cost flow [CMCF] problem which is the basis for the algorithms developed in this thesis.

The mathematical formulation of the CMCF problem is

$$\text{Min. } Z = \sum_{(i,j)\varepsilon A} c_{ij}\, x_{ij} \tag{2.1}$$

s.t.

$$\sum_{(j,i)\varepsilon I(i)} x_{ji} - \sum_{(i,j)\varepsilon O(i)} x_{ij} = \begin{cases} -V, \text{if } i=1 \\ V, \text{if } i=n, \ \forall\ i\varepsilon N \\ 0, \text{otherwise} \end{cases} \tag{2.2}$$

$$0 \le x_{ij} \le b_{ij}, \quad \forall\ (i,j)\ \varepsilon\ A \tag{2.3}$$

$$\sum_{(i,j)\varepsilon A} d_{ij}\, x_{ij} \le D \tag{2.4}$$

This algorithm is developed by Ahuja, Batra and Gupta [1]. The algorithm utilizes the concepts of parametric linear programming

and performs all the computations over the network. It first obtains the optimum basis corresponding to C_{min}, where C_{min} is the minimum value of C for which a feasible flow exists and then moves from one optimum basis to the next as D is decreased. The algorithm thus yields the minimum cost as a function of the budget, which is a piecewise linear convex function.

Optimality Conditions:

It is well known that the optimum basis of MCF problem is a spanning tree [5]. The addition of the budget constraint introduces one more arc in the optimum basis. It follows from bounded variable linear programming that the necessary and sufficient conditions for a feasible basis structure to be an optimum basis structure are that there exist dual variables π_j's and $\mu \geq 0$, satisfying the following conditions:

(i) $\pi_j - \pi_i = c_{ij} + \mu d_{ij}, \quad \forall (i,j) \varepsilon B$ (2.5)

(ii) $\pi_j - \pi_i \leq c_{ij} + \mu d_{ij}, \quad \forall (i,j) \varepsilon L$ (2.6)

(iii) $\pi_j - \pi_i \geq c_{ij} + \mu d_{ij}, \quad \forall (i,j) \varepsilon U$ (2.7)

(iv) $\mu \, (D - \sum_{(i,j)\varepsilon A} d_{ij} x_{ij}) = 0$ (2.8)

where B, L, and U represent the sets of arcs corresponding to the basic variables, the nonbasic variables at their lower bounds, and the nonbasic variables at their upper bounds,

respectively. The set B is called a basis and the triple (B, L, U) is called a basis structure.

Let $(T \cup \{(p,q)\}, L, U)$ be an optimum basis structure at $D = D'$. Further let π_j^c and π_j^d be the numbers satisfying

$$\pi_1^c = 0 \quad \text{and} \quad \pi_j^c - \pi_i^c = c_{ij}, \quad \forall\ (i,j) \in T \qquad (2.9)$$

$$\pi_1^d = 0 \quad \text{and} \quad \pi_j^d - \pi_i^d = d_{ij}, \quad \forall\ (i,j) \in T \qquad (2.10)$$

Define the numbers $\bar{c}_{ij}$ and $\bar{d}_{ij}$ for all arcs $(i,j) \in L \cup U$ as follows:

$$\bar{c}_{ij} = \begin{cases} \pi_i^c - \pi_j^c + c_{ij}, & \text{if } (i,j) \in L \\ \pi_j^c - \pi_i^c - c_{ij}, & \text{if } (i,j) \in U \end{cases} \qquad (2.11)$$

$$\bar{d}_{ij} = \begin{cases} \pi_i^d - \pi_j^d + d_{ij}, & \text{if } (i,j) \in L \\ \pi_j^d - \pi_i^d - d_{ij}, & \text{if } (i,j) \in U \end{cases} \qquad (2.12)$$

Clearly, $\bar{c}_{ij} = \bar{d}_{ij} = 0$, for all $(i,j) \in T$. If μ is a real number and π_j be the numbers satisfying,

$$\pi_1 = 0 \quad \text{and} \quad \pi_j - \pi_i = c_{ij} + \mu d_{ij}, \quad \forall\ (i,j) \in T$$

then,

$$\pi_j = \pi_j^c + \mu \pi_j^d, \qquad \forall\ j \in N \qquad (2.13)$$

substituting (2.13) in the conditions (2.5) - (2.8) and then using (2.11) and (2.12), we obtain the following equivalent conditions:

(i) $\bar{c}_{pq} + \mu \bar{d}_{pq} = 0$ (2.14)

(ii) $\bar{c}_{ij} + \mu \bar{d}_{ij} \geq 0 \quad \forall (i,j) \varepsilon L \cup U$ (2.15)

(iii) $\mu(D' - \sum_{(i,j)\varepsilon A} d_{ij} x_{ij}) = 0$ (2.16)

The conditions (2.14) - (2.16) are subsequently referred to as the optimality conditions. The $\bar{c}_{ij}$, $\bar{d}_{ij}$ and μ are referred to as the cost price, budget price and price ratio, respectively, associated with the basis structure $(T \cup \{(p,q)\}, L, U)$.

Depending upon the values of $\bar{c}_{ij}$ and $\bar{d}_{ij}$, arcs in $L \cup U$ can be classified as follows:

(i) active arcs: $\{(i,j) \varepsilon L \cup U: \bar{c}_{ij} > 0$ and $\bar{d}_{ij} < 0\}$.

(ii) critical arcs: $\{(i,j) \varepsilon L \cup U: \bar{c}_{ij} \leq 0$ and $\bar{d}_{ij} < 0\}$.

(iii) passive arcs: $\{(i,j) \varepsilon L \cup U: \bar{d}_{ij} \geq 0\}$.

The piecewise linear convex curve between cost and budget is obtained as follows.

Firstly an arc (p, q) belonging to the critical set is selected and added to basis. This results in the formation of exactly one cycle. Next the flow is augumented in this cycle. One of the arcs in the cycles reaches one of its bounds and leaves the basis respectively. The dual variables as well as (B, L, U) are updated and the step is repeated until the critical set is empty. We now describe how the algorithm moves from one basis to the next basis satisfying

the optimality conditions while decreasing the total budget D.

Addition of an arc $(i,j) \varepsilon L \cup U$ to the basic tree creates exactly one cycle W_{ij} consisting of the basic arcs. We define the orientation of the cycle W_{ij} along (i,j) if $(i,j) \varepsilon L$ and opposite to (i,j) if $(i,j) \varepsilon U$. Let $\bar{W}_{ij}$ and $\underline{W}_{ij}$ be the sets of arcs in the cycle W_{ij} along and opposite to its orientation, respectively. Then using (2.9) and (2.10) it can be easily shown that

$$\bar{c}_{ij} = \sum_{(i,j)\varepsilon\bar{W}_{ij}} c_{ij} - \sum_{(i,j)\varepsilon\underline{W}_{ij}} c_{ij} \tag{2.17}$$

$$\bar{d}_{ij} = \sum_{(i,j)\varepsilon\bar{W}_{ij}} d_{ij} - \sum_{(i,j)\varepsilon\underline{W}_{ij}} d_{ij} \tag{2.18}$$

Thus, $\bar{c}_{ij}$ (or $\bar{d}_{ij}$) denotes the increase in the cost (or budget) if unit amount of additional flow is circulated in the cycle W_{ij} along its orientation. The above given classification of non-tree arcs can be given the physical interpretation. Active arcs are those arcs which can lead to increase in cost if budget is decreased. Critical arcs are those which do not increase cost if budget is reduced. Passive arcs do not lead to decrease in budget even if more cost occurs.

Characteristic Interval:

Let at $D = \underline{D}$, the optimum basis structure of the CMCF problem is $(T \cup \{(p,q)\}, L, U)$. Let $\underline{x}_{ij}$ be the flow on arc $(i,j) \varepsilon A$ and $\underline{Z}$ be the cost of this flow. Further, let

c_{ij}, $\bar{d}_{ij}$ and be the cost prices, budget prices and price ratio associated with the given basis structure.

We now determine the interval $(\underline{D}, \bar{D})$ for the values of D for which the given basis structure continues to remain optimum. This interval is known as the characteristic interval associated with $(T \cup \{(p,q)\}, L, U)$.

Since the numbers, $\bar{c}_{ij}$, $\bar{d}_{ij}$ and are uniquely determined for a given basis structure, the optimality conditions (2.14) and (2.15) are not affected by decrease in the value of D. However, since $\lambda L > 0$, the flow must be changed in order to satisfy (2.16). The only way to change the flow, without changing the basis structure and satisfying the flow conservation constraints (2.2), is by circulating some additional flow in the cycle W_{pq} along its orientation. It was noted that $\bar{d}_{pq} < 0$ is the rate at which the budget is reduced and $\bar{c}_{pq} > 0$ is the rate at which the additional cost is incurred when unit amount of flow is circulated. Since the changed flow must also satisfy the bound restrictions of the arcs (2.3), we calculate the maximum flow $\bar{f}$ that can be circulated without violating the bound restrictions of the arcs in W_{pq}. If we define,

$$\bar{f}_{ij} = \begin{cases} b_{ij} - \underline{x}_{ij}, & \text{if } (i,j) \in \bar{W}_{pq} \\ \underline{x}_{ij}, & \text{if } (i,j) \in \underline{W}_{pq} \end{cases}$$

then,

$$\bar{f} = \min_{(i,j)\varepsilon W_{pq}} \{f_{ij}\}$$

Thus,

$$\bar{D} = \underline{D} + \bar{f}\,\bar{d}_{pq}$$

For all values of D in $(\underline{D}, \bar{D})$, the optimum flow is given by,

$$x_{ij} = \begin{cases} \underline{x}_{ij} + \alpha'\bar{f}\,, & \text{if } (i,j)\ \varepsilon\ \bar{W}_{pq} \\ \underline{x}_{ij} - \alpha'\bar{f}\,, & \text{if } (i,j)\ \varepsilon\ \underline{W}_{pq} \\ x_{ij}\,, & \text{otherwise} \end{cases}$$

and cost by

$$Z = \underline{Z} + \alpha'\bar{f}\,\bar{c}_{pq},$$

where, $\alpha' = (D - \underline{D})/(\bar{D} - \underline{D})$

Let $\bar{x}_{ij}$ denote the flow in arc $(i,j)\ \varepsilon\ A$ at $D = \bar{D}$. If it is required to find the optimum flow for $D < \bar{D}$, then a dual simplex iteration is performed to obtain a new basis structure at $D = \bar{D}$.

The Dual Simplex Iteration:

At $D = \bar{D}$, flow in an arc $(u, v)\ \varepsilon\ W_{pq}$, for which $\bar{f}_{uv} = \bar{f}$, equals its lower or upper bound. If D is decreased further, the bound is violated. Thus, to obtain a new basis structure at $D = \bar{D}$, which may permit decrease in the value of D, the arc (u, v) is dropped from the basis and a non-basic arc

is selected to enter the basis. The arc (u, v) becomes a non-basic arc at its respective bound. In the new basis structure, the dashed values and sets represent the corresponding values and sets of the previous basis structure. By selecting an active arc and performing dual simplex iteration each time until the active set is empty, the cost is increased and the budget is reduced.

The curve starts at C_{min} and as the value of D is decreased, the value of Z keeps increasing until a value of D is reached when Z stops increasing. The slope of the curve at any point is the value of $-\mu$ in the optimum solution which corresponds to that point. Since the value of $-\mu$ keeps increasing and finally becomes zero and no critical arcs are formed at any iteration (The proof is given in [1]), the curve is a piecewise linear convex function.

2.5 An Overview of Goal Programming:

As mentioned in the previous section, Goal programming allows the decision maker to specify a target for each objective function which provides him the preferred solution by minimizing the sum of the deviations from the prescribed set of target values.

Generalized goal programming has a number of special terms and concepts that are being used in this thesis. They are mentioned as below:

(i) Objective:

An objective is a relatively general statement (in narrative or quantitative terms) that reflects the desires of the decision maker. For example, one may wish to "maximize profit" or "minimize total wastage" or "wipe out poverty".

(ii) Aspiration Level:

An aspiration level is a specific value associated with the desired or acceptable level of achievement of an objective. Thus, an aspiration level is used to measure the achievement of an objective and generally serves to "anchor" the objective to reality.

(iii) Goal:

An objective in conjunction with an aspiration level is termed a goal. For example, we may wish to "achieve at least X units of profits" or "reduce the rate of inflation by Y percent."

(iv) Goal Deviation:

The difference between what one accomplishes and what one aspires to is the deviation from his goal. A deviation can represent over as well as under achievement of a goal.

(v) Goal Formulation:

We will now examine how to mathematically transform an objective into a goal within our goal programming frame work.

Consider the objective function expressed in general terms as $f_i(X)$. The procedure to be presented is applicable whether $f_i(X)$ is linear or nonlinear, but only linear objectives are considered in this thesis.

We then let,

$f_i(X)$ = mathematical representation of objective i as a function of the decision variables $X = (x_1, x_2, \ldots, x_n)$

b_i = value of the aspiration level associated with objective i.

Three possible forms of goals may then result:

(i) $f_i(X) \leq b_i$: that is, we wish to have a value of $f_i(X)$ that is equal to or less than b_i.

(ii) $f_i(X) \geq b_i$: that is, we wish to have a value of $f_i(X)$ that is equal to or greater than b_i.

(iii) $f_i(X) = b_i$: that is $f_i(X)$ must exactly equal b_i.

Regardless of the form, we shall transform any of these relations into the goal programming format by adding

a negative deviation variable ($\beta_i \geq 0$) and subtracting a a positive deviation variable ($\alpha_i \geq 0$). This statement is summarized as below:

Goal Type	Goal Programming Form	Deviation Variables to be Minimized
$f_i(X) \leq b_i$	$f_i(X) + \beta_i - \alpha_i = b_i$	α_i
$f_i(X) \geq b_i$	$f_i(X) + \beta_i - \alpha_i = b_i$	β_i
$f_i(X) = b_i$	$f_i(X) + \beta_i - \alpha_i = b_i$	$\alpha_i + \beta_i$

We will briefly discuss the three extensions of goal programming, They are:

(1) Weighted goal programming.

(2) Interval goal programming.

(3) Fuzzy goal programming.

<u>Weighted Goal Programming:</u>

In this model the decision maker assigns an aspiration level for each of the objectives and also the weighting factors for each of the deviations. It obtains an optimum solution by minimizing the sum of weighted deviations.

There is an alternative way in which the weighted model may be formed. Rather than multiplying each deviation variable by a constant weight, we may instead, raise each

deviation variable in the achievement function to some power. This results in a polynomial form for the achievement function.

Given a multi-objective model,

optimize Z_i $\quad i = 1, \ldots, s$

s.t.

$$AX \leq b$$
$$X \geq 0$$

Adding aspiration levels and deviations variables to each objective and weighting each resultant goal, we obtain,

$$\text{Minimize} \quad a = \sum_{i=1}^{s} (w_i \alpha_i + r_i \beta_i)$$

subject to

$$Z_i(X) + \beta_i - \alpha_i = Z_i^o, \quad i = 1, \ldots, s$$
$$AX \leq b$$
$$X, \alpha_i, \beta_i \geq 0, \quad i = 1, \ldots, s$$

where,

α_i = Weighting factor for the positive deviation of goal i.

β_i = Weighting factor for the negative deviation of goal i.

Z_i^o = Aspiration level for objective i.

Interval Goal Programming:

In this model the decision maker specifies a range of aspiration levels for each of the goals instead of one

aspiration level. It obtains an optimum solution by minimizing the weighted sum of deviations from the set of ranges.

The mathematical formulation of this model is,

$$\text{Minimize} \quad a = \sum_{i=1}^{s} (w_{i,2}\,\alpha_{i,2} + r_{i,1}\,\beta_{i,1})$$

subject to

$$Z_{i,1} + \beta_{i,1} - \alpha_{i,1} = L_i \qquad \text{all } s$$

$$Z_{i,2} + \beta_{i,2} - \alpha_{i,2} = U_i \qquad \text{all } s$$

where,

$Z_{i,1} = Z_{i,2}$ = expression for objective k.

$\beta_{i,1}, \beta_{i,2}$ = negative deviations.

$\alpha_{i,1}, \alpha_{i,2}$ = positive deviations.

$r_{i,1}$ = weighting factor for the negative deviation for goal $Z_{i,1}$.

$w_{i,2}$ = weighting factor for the positive deviation for goal $Z_{i,2}$.

Fuzzy Linear Programming.

A fairly recent attempt at modeling and solving the multiple-objective problem is that known as fuzzy programming. The approach is similar, in many respects, to the weighted linear goal programming method previously discussed, differing primarily in the manner in which the importance of the goals

are considered. This method minimizes the worst under achievement of any goal.

A major advantage of fuzzy linear programming is that it may be transformed into a conventional linear programming model. The main disadvantage of this method is, the under-achievement of just one goal can have a major impact on the solution, since it attempts to minimize the maximum under-achievement.

CHAPTER III

NETWORK GOAL PROGRAMMING ALGORITHMS

3.1 Introduction:

In this chapter, we shall consider two special classes of linear goal programming problems, i.e. weighted goal network flow problem and interval goal network flow problem.

We generally confront problems that require two objectives to be considered simultaneously. Often, these objectives are conflicting in nature. These types of problems may arise in road networks, communication networks and pipe line networks. The typical applications of bi-criteria network flow problems are mentioned in Section 1.1.

One of the methods of solving bi-criteria problems is to obtain the efficient or non-dominated solutions set. As the efficient solution set is considerably large, it becomes practically difficult for the decision maker to choose the solution he would prefer.

Goal programming is a technique which takes the decision maker's preference into consideration and provides him the preferred solution. He expresses his preference by specifying targets to the objectives and the weights to the deviations from the targets.

In weighted goal programming, the decision maker specifies a target for each of the objective and weights for the deviations. It provides him an optimum solution which is as close as possible to the specified targets, by minimizing the sum of weighted deviations.

In interval goal programming the decision maker specifies a range or interval for each of the objectives and weights for each of the deviations from lower and upper bounds of the interval. It obtains an optimum solution lying in the ranges specified if one exists or finds a solution such that the sum of weighted deviations are minimum.

This chapter has been divided into 11 sections. In Sec. 2 we present the notations used to represent the feasible region of the bi-criteria network flow problem. The tracing of various trade-off curves is presented in Sec. 3. The mathematical formulation and development of the WGNF algorithm are presented in Sec. 4 and 5. In Sec. 6 and Sec. 7 we present the statement of the WGNF algorithm and a numerical example. The mathematical formulation and development of IGNF algorithm are given in Sec. 8 and Sec. 9. A numerical example of IGNF problem is given in Sec. 10. Finally the computational performance of both the algorithms is reported in Sec. 11.

3.2 Notations:

The various regions as shown in Fig. 3.1 are defined as below:

E : $\{(x,y) : x \geq 0 \text{ and } y \geq 0 \}$.

S : $\{(x,y) : x = \sum_{(i,j)\varepsilon A} c_{ij}\, x_{ij},\ \ y = \sum_{(i,j)\varepsilon A} d_{ij}\, x_{ij}\}$.

R_1: $\{(x,y) : 0 \leq x \leq f_1,\ 0 \leq y \leq g_2 \text{ and } (x,y) \notin S\}$

R_2: $\{(x,y) : 0 \leq x \leq f_3,\ g_2 \leq y \leq \infty \text{ and } (x,y) \notin S\}$

R_3: $\{(x,y) : f_3 \leq x \leq \infty,\ g_4 \leq y \leq \alpha \text{ and } (x,y) \notin S\}$

R_4: $\{(x,y) : f_1 \leq x \leq \infty,\ 0 \leq y \leq g_4 \text{ and } (x,y) \notin S\}$

S_1: $\{(x,y) \varepsilon S : x \leq f_1\}$

S_2: $\{(x,y) \varepsilon S : x \leq f_3\}$

S_3: $\{(x,y) \varepsilon S : x > f_3\}$

S_4: $\{(x,y) \varepsilon S : x > f_1\}$

α_i: positive deviation from goal i.

β_i: negative deviation from goal i.

w_i: weight assigned to positive deviation α_i.

r_i: weight assigned to negative deviation β_i.

$w_i,\ r_i > 0$.

3.3 Tracing the Trade-Off Curves:

The optimum solutions for WGNF problem and IGNF problem lie either on B_1, B_2, B_3 and B_4 curves or in the feasible region S as shown in Fig. 3.1. Since our algorithm searches for an optimum solution by traversing along $B_1 \cup B_2 \cup B_3 \cup B_4$

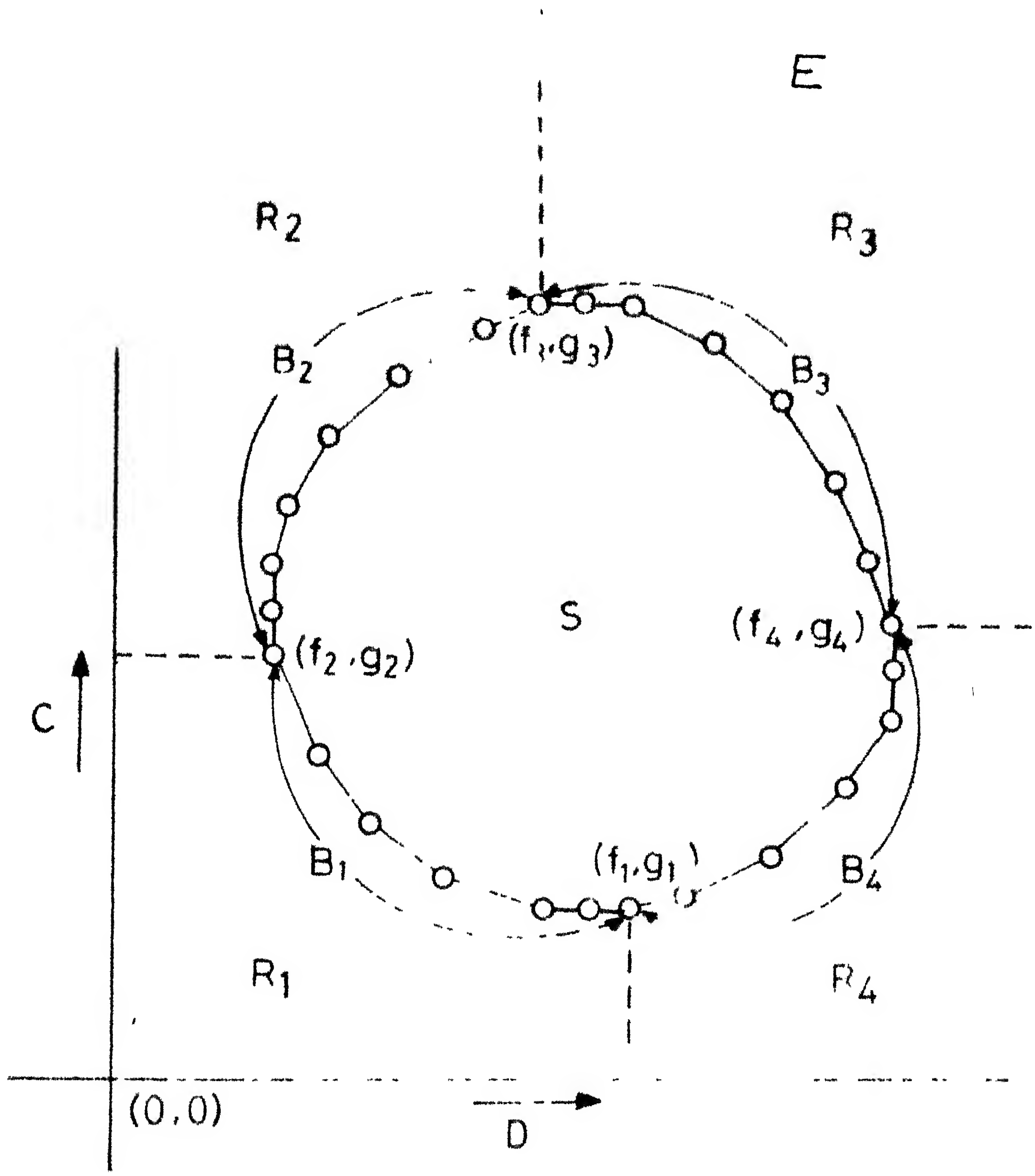

Fig. 3.1 Feasible region of bicriteria network flow problem

and through the feasible region S it is necessary to trace the B_1, B_2, B_3 and B_4 curves. Once these curves are obtained, it can be shown that they to-gether form a closed convex feasible region S.

Tracing B_1 Curve:

The detailed procedure for obtaining the B_1 curve was presented in Sec. 2.4. The procedure for tracing B_2, B_3 and B_4 curves is same as that for tracing B_1, except that the optimality conditions, the critical set, the active set, the passive set and the criteria for selecting the price ratio differ, as the problems considered to obtain each of these curves are different.

Tracing B_2 Curve:

B_2 curve is a piecewise linear concave function between cost and budget. This curve is obtained by parametrically solving the following problem.

$$\text{Max.} \quad Z = \sum_{(i,j)\varepsilon A} c_{ij} \, x_{ij}$$

subject to

$$\sum_{(j,i)\varepsilon I(i)} x_{ji} - \sum_{(i,j)\varepsilon O(i)} x_{ij} = \begin{cases} -V, & \text{if } i=1 \\ V, & \text{if } i=n, \forall \; i\varepsilon N \\ 0, & \text{otherwise} \end{cases}$$

$$0 \leq x_{ij} \leq b_{ij} \qquad \forall \; (i,j) \; \varepsilon \; A$$

$$\sum_{(i,j)\varepsilon A} d_{ij} \, x_{ij} < D$$

The optimality conditions are:

(i) $\bar{c}_{pq} - \mu \bar{d}_{pq} = 0$

(ii) $-\bar{c}_{ij} + \mu \bar{d}_{ij} \geq 0 \quad \forall \; (i,j) \in L \cup U$

(iii) $\mu(D - \sum_{(i,j)\in A} d_{ij} \, x_{ij}) = 0$

The non-basic arcs are classified as:

(i) Active arcs S' : $\{(i,j) \in L \cup U: \bar{c}_{ij} < 0 \text{ and } \bar{d}_{ij} < 0\}$

(ii) Critical arcs S": $\{(i,j) \in L \cup U: \bar{c}_{ij} \geq 0 \text{ and } d_{ij} < 0\}$

(iii) Passive arcs S''': $\{(i,j) \in L \cup U: \bar{d}_{ij} \geq 0\}$

Initially, $\sum_{(i,j)\in A} c_{ij} \, x_{ij}$ is maximized and let (C_{max}, D) be the solution. A critical arc $(p,q) \in S''$ is selected and entered into the basis and dual-simplex iteration is performed. This is continued until S is empty.

Next, an active arc (p,q) that that $\mu_{pq} = \min_{(i,j)\in S'} (\mu_{ij})$ is selected and entered into the basis and dual-simplex iteration is performed. Each dual-simplex iteration results in an extreme solution. This is repeated until $S' = \{\emptyset\}$. As μ is gradually increasing and ultimately becomes zero along with gradual decrease in C and D, the curve B_2 is a piecewise concave linear function.

Tracing B_3 Curve.

B_3 is a piecewise linear concave function between cost and budget. This curve is obtained by parametrically solving the

following problem.

$$\max. \quad Z = \sum_{(i,j)\varepsilon A} c_{ij}\, x_{ij}$$

subject to

$$\sum_{(j,i)\varepsilon I(i)} x_{ji} - \sum_{(i,j)\varepsilon O(i)} x_{ij} = \begin{cases} -V, & \text{if } i=1 \\ V, & \text{if } i=n, \ \forall\ i\varepsilon N \\ 0, & \text{otherwise} \end{cases}$$

$$0 \leq x_{ij} \leq h_{ij} \qquad \forall\ (i,j)\ \varepsilon\ A$$

$$\sum_{(i,j)\varepsilon A} d_{ij}\, x_{ij} \geq D$$

The optimality conditions are:

(i) $\bar{c}_{pq} + \mu\, \bar{d}_{pq} = 0$

(ii) $-\bar{c}_{ij} - \mu \bar{d}_{ij} \geq 0 \qquad \forall\ (i,j)\ \varepsilon\ A$

(iii) $\mu\left(\sum_{(i,j)\varepsilon A} d_{ij}\, x_{ij} - D\right) = 0$

The non basic-arcs are classified as:

(i) Active arcs S' : $\{(i,j)\ \varepsilon\ L \cup U\colon \bar{c}_{ij} <$ and $\bar{d}_{ij} > 0\}$

(ii) Critical arcs S'': $\{(i,j)\ \varepsilon\ L \cup U\colon \bar{c}_{ij} \geq 0$ and $\bar{d}_{ij} > 0\}$

(iii) Passive arcs S''': $\{(i,j)\ \varepsilon\ L \cup U\colon \bar{d}_{ij} \leq 0\}$

Initially, $\sum_{(i,j)\varepsilon A} c_{ij}\, x_{ij}$ is maximized and let (C_{max}, D) be the solution. Next, the critical arc set is emptied by performing dual-simplex iterations. An active arc (p,q) such that

$\mu_{pq} = \min_{(i,j)\varepsilon S} (\mu_{ij})$ is selected and entered into the basis.

The dual simplex iteration is performed. This is repeated until

$S' = \{\phi\}$. Since the value of μ is gradually increasing and C is gradually decreasing with the increase in D, the curve B_3 is a piecewise concave function.

Tracing B_4 Curve:

B_4 curve is a piecewise linear convex function between cost and budget. This curve is obtained by parametrically solving the following problem:

$$\text{Min.} \quad Z = \sum_{(i,j)\varepsilon A} c_{ij} \, x_{ij}$$

s.t.

$$\sum_{(j,i)\varepsilon I(i)} x_{ji} - \sum_{(i,j)\varepsilon O(i)} x_{ij} = \begin{cases} -V, & \text{if } i=1 \\ V, & \text{if } i=n, \ \forall \ i \ \varepsilon \ N \\ 0, & \text{otherwise} \end{cases}$$

$$0 \leq x_{ij} \leq b_{ij} \qquad \forall \ (i,j) \ \varepsilon \ A$$

$$\sum_{(i,j)\varepsilon A} d_{ij} \, x_{ij} \geq D$$

The optimality conditions are.

(i) $\quad \bar{c}_{pq} - \mu \bar{d}_{pq} = 0$

(ii) $\quad \bar{c}_{ij} - \mu \bar{d}_{ij} > 0 \qquad \forall \ (i,j) \ \varepsilon \ L \cup U$

(iii) $\quad \mu \left(\sum_{(i,j)\varepsilon A} d_{ij} \, x_{ij} - D \right) = 0$

The non-basic arcs are classified as:

(i) Active arcs S' : $\{(i,j) \ \varepsilon \ L \cup U : \bar{c}_{ij} > 0 \text{ and } \bar{d}_{ij} > 0\}$

(ii) Critical arcs S'': $\{(i,j) \ \varepsilon \ L \cup U : \bar{c}_{ij} \leq 0 \text{ and } \bar{d}_{ij} > 0\}$

(iii) Passive arcs S''': $\{(i,j) \varepsilon L \cup U : \bar{d}_{ij} \leq 0\}$

Initially $\sum_{(i,j)\varepsilon A} c_{ij} x_{ij}$ is minimized and let (C_{min}, D) be the solution. Next, the critical arc set is emptied by performing dual-simplex iterations. An active arc (p,q) such that $\mu_{pq} = \min_{(i,j)\varepsilon S'}. (\mu_{ij})$ is selected and entered into the basis. The dual simplex iteration is performed. This is repeated until $S' = \{\phi\}$. A dual simplex iteration results in an extreme solution. Since the value of μ is gradually increasing and both C and D are increasing, the curve B_4 is a piecewise convex function.

The curves B_1 and B_4 together constitute a convex curve as the starting solution is same and the cost is gradually increasing. The curves B_2 and B_3 together constitute a concave curve as the starting solution is the same and the cost is gradually decreasing. It follows from the above discussion that the terminating solution of B_1 is the starting solution for B_2 and the terminating solution of B_2 is the starting solution for B_3. Similarly the terminating solution of B_3 is the starting solution for B_4 and the terminating solution of B_4 is the starting solution for B_1. Thus we obtain a closed region enclosed by the curves $B_1 \cup B_2 \cup B_3 \cup B_4$. Since $B_1 \cup B_4$ is a convex curve and $B_2 \cup B_3$ is a concave curve, the closed feasible region is a closed convex region.

3.4 Mathematical Formulation of the Weighted Goal Network Flow Problem:

In this section, we give the mathematical formulation of the weighted goal network flow problem.

In the network $G = (N, A)$, associated with each arc $(i,j) \varepsilon A$ are two numbers c_{ij} and d_{ij}. The capacity of each arc $(i,j) \varepsilon A$ is b_{ij}. The amount that is to be shipped from source 1 to sink n is V.

$$\text{Minimize} \quad Z = w_1\alpha_1 + r_1\beta_1 + w_2\alpha_2 + r_2\beta_2 \tag{3.1}$$

subject to the following constraints:

Flow Conservation Constraints:

These constraints essentially represent the factk that flow of the commodity is conserved at all nodes, except at source and sink.

$$\sum_{(j,i)\varepsilon I(i)} x_{ji} - \sum_{(i,j)\varepsilon O(i)} x_{ij} = \begin{cases} -V, & \text{if } i=1 \\ V, & \text{if } i=n, \\ 0, & \text{otherwise} \end{cases} \quad \forall\, i \;\varepsilon\; N \tag{3.2}$$

Capacity Constraints:

These constraints essentially represent the fact that flow over an arc (i,j) cannot exceed its capacity

$$0 \le x_{ij} \le b_{ij}, \quad \forall\, (i,j) \;\varepsilon\; A \tag{3.3}$$

Goal Constraints:

An objective in conjunction with an aspiration level and deviations is known as goal constraint.

$$\sum_{(i,j)\varepsilon A} c_{ij}\, x_{ij} + \beta_1 - \alpha_1 = C_1 \qquad (3.4)$$

$$\sum_{(i,j)\varepsilon A} d_{ij}\, x_{ij} + \beta_2 - \alpha_2 = D_1 \qquad (3.5)$$

3.5 Development of the Algorithm:

We shall now discuss, the various steps involved in finding an optimum solution for a specified (C_1, D_1, w_1, r_1, w_2, r_2). Let us consider (C_1, D_1) ε R_2 in order to trace all the possible steps of the WGNF algorithm.

Initially, the minimum cost flow problem with $\sum_{(i,j)\varepsilon A} c_{ij}\, x_{ij}$ as an objective function is solved. Let (C_{min}, D') be the minimum cost and budget obtained and it is represented by I in Fig. 3.3. The deviations axes from aspiration levels (C_1, D_1) are shown in Fig. 3.2. If the algorithm traverses in quadrant I then it minimizes α_1 and α_2 and the other two deviations are zero. Similarly α_1 and β_2 are minimized in quadrant II, β_2 and β_1 are minimized in quadrant III and α_2 and β_1 are minimized in quadrant IV. Therefore, the algorithm always minimizes only two deviations and the other two being at zero. The above statement is supported by theorems 1 to 5 given below.

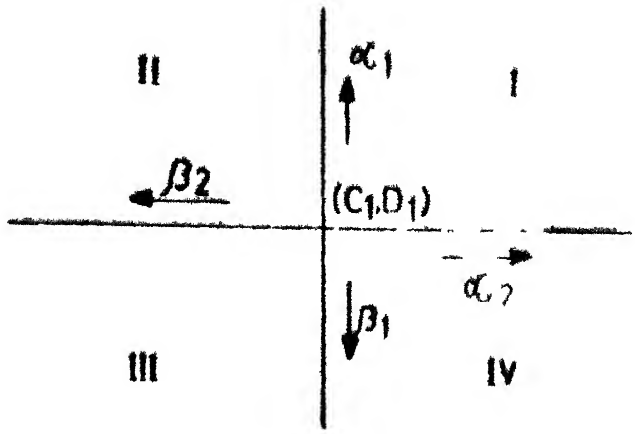

Fig. 3·2 Aspiration levels and devaiations for WGNF problem

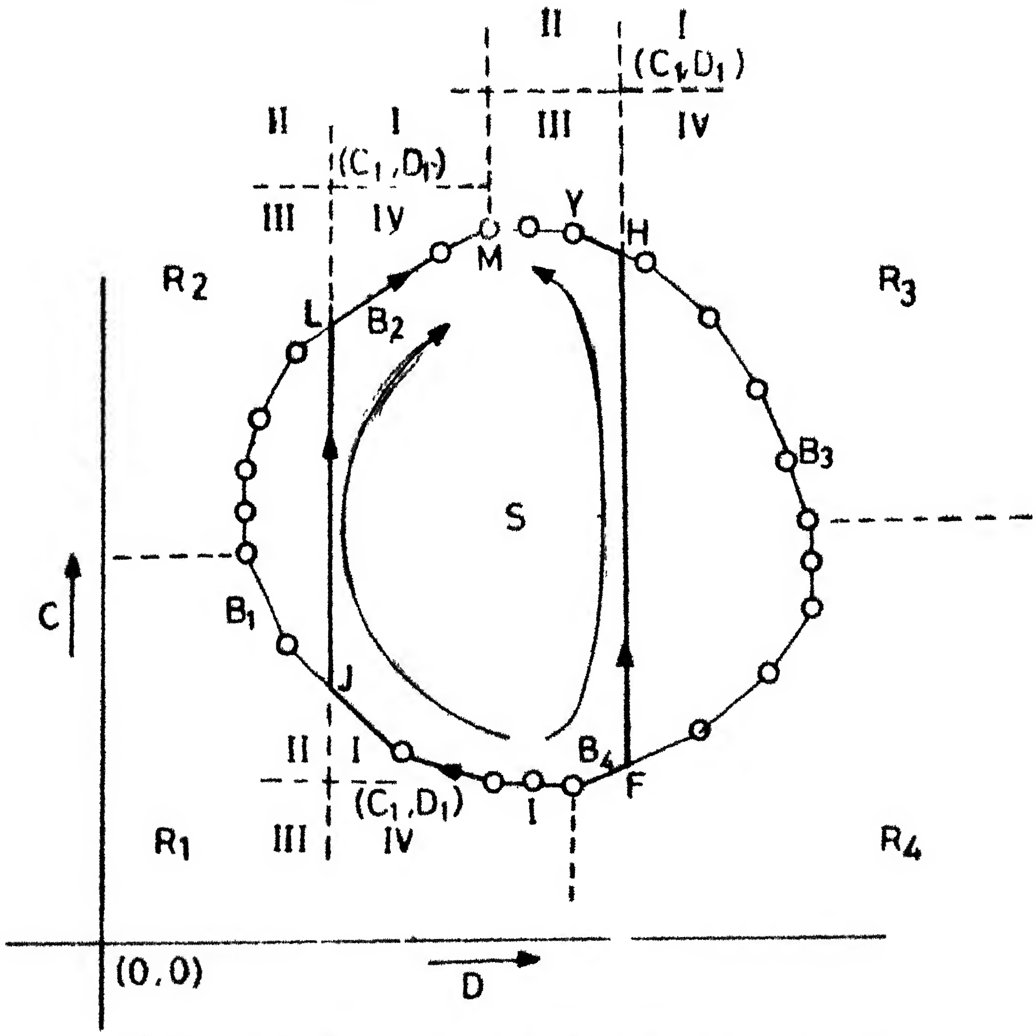

Fig. 3·3 Paths traced by WGNF regorithm to obtain optimum solutions

THEOREM 1: $\alpha_i \beta_i = 0, \forall \; i = 1,2$

Proof: We will prove this theorem by contradiction. Suppose $(X^*, \alpha_1^*, \beta_1^*)$ be an optimum solution and $\alpha_1^*, \beta_1^* > 0$. Let $C^* = \sum_{(i,j)\in A} c_{ij} x_{ij}^*$

The value of the objective function is

$$Z_1 = w_1\alpha_1^* + r_1\beta_1^* \tag{3.6}$$

The goal constraint is

$$C^* + \beta_1 - \alpha_1 = C_1 \tag{3.7}$$

Let $\Delta = \min. (\beta_1^*, \alpha_1^*)$

Consider the solution $(X^*, \beta_1', \alpha_1')$,

let, $$\beta_1' = \beta_1^* - \Delta \tag{3.8}$$

$$\alpha_1' = \alpha_1^* - \Delta \tag{3.9}$$

substituting (3.8) and (3.9) in (3.6) and (3.7), we get,

$$Z_2 = w_1(\alpha_1' + \Delta) + r_1(\beta_1' + \Delta)$$

$$= w_1\alpha_1' + r_1\beta_1' + \Delta(w_1 + r_1) \tag{3.10}$$

goal constraint is

$$C^* + \beta_1' - \alpha_1' = C_1 \tag{3.11}$$

Hence the goal constraint is satisfied.

Since Δ, w_1, $r_1 > 0$ it follows that $Z_2 > Z_1$. This contradicts that $(X^*, \beta_1^*, \alpha_1^*)$ is an optimum solution. Therefore $\Delta = 0$. Hence the theorem follows.

Similarly it can be proved for $\alpha_2 \beta_2 = 0$. Therefore atleast two of α_1, β_1, α_2 and β_2 are zero.

QED

THEOREM 2: If $(C_1, D_1) \varepsilon R_1$, then $(C^*, D^*) \varepsilon B_1$ and $D^* \geq D_1$ and $C^* \geq C_1$.

Proof: Consider $(C', D') \varepsilon S_2$. There exists $(C', D) \varepsilon S_1$, such that $D < D'$. Since $D < D'$, it follows that $\alpha_2 < \alpha_2'$ resulting in $Z < Z'$ where Z is the value of objective function (C', D). Hence a better solution exists in S_1 for every solution in S_2.

Consider a feasible solution $(C', D') \varepsilon S_1$. A feasible solution $(C, D') \varepsilon B_1$ exists such that $C < C'$. Since $C < C'$ it follows that $\alpha_1 < \alpha_1'$. It results in $Z < Z'$ where Z and Z' are objective functions of (C, D') and (C', D') respectively. Therefore for any solution in S, a better solution exists on B_1.

Let $(C', D') \varepsilon B_1$ and $D' < D_1$. There exists $(C, D_1) \varepsilon B_1$ and $C < C'$. Therefore $\alpha_1 < \alpha_1'$ resulting in $Z < Z'$.

Let $(C'', D'') \varepsilon B_1$ and $C'' < C_1$. There exists (C_1, D) and $C_1 > C''$. Therefore $\alpha_2 < \alpha_2''$ and $\beta_1 < \beta_1''$, resulting $Z < Z''$. Hence an optimum solution is $(C^*, D^*) \varepsilon B_1$ and $D^* \geq D$, and $C^* \geq C_1$. It follows from theorem 1 and above that if $(C_1, D_1) \varepsilon R_1$ then the optimum solution is $(X^*, \alpha_1^*, \alpha_2^*)$ and β_1, $\beta_2 = 0$.

Similarly we can prove the following theorems.

QED

THEOREM 3: If $(C_1, D_1) \varepsilon R_2$ then $(C^*, D^*) \varepsilon B_2$ and $D^* \geq D_1$ and $C^* \leq C_1$.

The optimum solution is $(X^*, \alpha_2^*, \beta_1^*)$ and $\alpha_1, \beta_2 = 0$.

THEOREM 4: If $(C_1, D_1) \varepsilon R_3$, then $(C^*, D^*) \varepsilon B_3$ and $D^* \leq D_1$ and $C^* \leq C_1$.

The optimum solution is $(X^*, \beta_1^*, \beta_2^*)$ and $\alpha_1, \alpha_2 = 0$.

THEOREM 5: If $(C_1, D_1) \varepsilon R_4$, then $(C^*, D^*) \varepsilon B_4$ and $D^* \leq D_1$ and $C^* \geq C_1$.

The optimum solution is $(X^*, \beta_2^*, \alpha_1^*)$ and $\alpha_2, \beta_1 = 0$.

QED

If $D_1 \leq D'$ then the algorithm moves along the B_1 curve in the direction of J as shown in Fig. 3.3. Otherwise it traverses along B_4. The algorithm evaluates the objective function values of extreme-solutions and two non-extreme solutions, namely $(C, D_1) \varepsilon B_1 \cup B_2 \cup B_3 \cup B_4$ and $(C_1, D) \varepsilon B_1 \cup B_2 \cup B_3 \cup B_4$, because one of these solutions is optimum if $(C_1, D_1) \not\varepsilon S$. We ascertain the above statement in theorems 6 to 9 given below.

THEOREM 6: If $(C_1, D_1) \not\varepsilon S$, then one of the extreme solutions of $B_1 \cup B_2 \cup B_3 \cup B_4$ is optimum.

Proof: Consider two adjacent extreme solutions (C', D') and (C'', D''). Any non-extreme solution $(\bar{C}, \bar{D})$ lying between (C', D') and (C'', D'') can be expressed as convex combination of

(C', D') and (C'', D''). Therefore Z' or Z'' is atleast as good as any $\bar{Z}$, depending on $Z' \leq Z''$ or $Z'' \leq Z'$. Hence the theorem follows.

QED

THEOREM 7: If $(C_1, D) \in B_1$ lies between two adjacent extreme solutions $(C', D') \in B_1$ and $(C'', D'') \in B_1$ such that $C_1 > C''$, $D' > D_1$ and satisfying $Z' \geq Z$ where Z', Z and Z'' are objective function values of (C', D'), (C_1, D) and (C'', D'') respectively, then (C_1, D) is optimum.

Proof: $$Z'' = w_2\alpha_2'' + r_1\beta_1'' \tag{3.12}$$

$$Z = w_2\alpha_2$$

Since $\alpha_2 < \alpha_2''$, it follows that $Z < Z''$. (C_1, D) is atleast as good as any other solution lying between (C_1, D) and (C', D') as they can be expressed as a convex combination of (C_1, D) and (C', D') and $Z' \geq Z$. Similarly (C_1, D) is better than any other solution lying between (C_1, D) and (C'', D'') because they can be expressed as a convex combination of (C_1, D) and (C'', D'') and $Z'' > Z$. According to theorem 2, for any solution in S, a better solution exists on B_1. Hence (C_1, D) is atleast as good as any neighbouring solution. Therefore (C_1, D) is a local optima. In linear programming, local optima is also a global optima. Hence the theorem follows.

Similarly this type of result can be proved for each of $(C_1, D) \varepsilon B_4$, $(C, D_1) \varepsilon B_2$ and $(C, D_1) \varepsilon B_3$ lying between two adjacent extreme solutions, (C', D') and (C'', D'') satisfying $Z' \geq Z$ then they are optimum.

QED

THEOREM 8: If $(C, D_1) \varepsilon B_1$ lies between two adjacent extreme solutions $(C', D') \varepsilon B_1$ and $(C'', D'') \varepsilon B_1$ such that $C > C''$, $C'' > C_1$ and satisfying $Z < Z''$ where Z, Z', Z'' are objective function values of (C, D_1), (C', D') and (C'', D'') respectively, then (C, D_1) is optimum.

Proof: $Z' = w_1\alpha_1' + r_2\beta_2'$

$Z = w_1\alpha_1$

Since $\alpha_1 < \alpha_1'$, it follows that $Z < Z'$. (C, D_1) is a better than any solution between (C', D') and (C, D_1) and between (C, D_1) and (C'', D'') as they can be expressed as convex combination of these solutions and $Z < Z'$ and $Z < Z''$. According to theorem 2, for any solution in S, a better solution exists on B_1. Hence (C, D_1) is atleast as good as any neighbouring solution. Therefore (C, D_1) is a local optima. In linear programming, local optima is also a global optima. Hence the theorem follows.

Similarly this type of result can be proved for each of $(C, D_1) \varepsilon B_4$, $(C_1, D) \varepsilon B_2$ and $(C_1, D) \varepsilon B_3$ lying between two

adjacent extreme solutions, (C', D') and (C'', D'') satisfying $Z < Z''$ then they are optimum.

QED

THEOREM 9: If $(C, D_1) \varepsilon B_1$ and $(C_1, D) \varepsilon B_1$ lie between two adjacent extreme solutions $(C', D') \varepsilon B_1$ and $(C'', D'') \varepsilon B_1$ such that $C_1 > C''$, then

(i) (C_1, D) is optimum if $\underline{\underline{Z}} \leq \bar{\bar{Z}}$ and

(ii) (C, D_1) is optimum if $\underline{\underline{Z}} \geq \bar{\bar{Z}}$

where $\underline{\underline{Z}}$ and $\bar{\bar{Z}}$ are the objective function values of (C_1, D) and (C, D_1).

Proof: We shall prove case (i).

Let Z' and Z'' be objective function values of (C', D') and (C'', D'')

$$Z'' = w_2\alpha_2'' + r_1\mu_1''$$

$$\underline{\underline{Z}} = w_2\underline{\underline{\alpha}}_2$$

Since $\underline{\underline{\alpha}}_2 < \alpha_2''$, it follows that $\underline{\underline{Z}} < Z''$. (C_1, D) is atleast as good as any solution lying between (C_1, D) and (C, D_1) because they can be expressed as convex combination of (C_1, D) and (C, D_1) and $\underline{\underline{Z}} \leq \bar{\bar{Z}}$. Similarly (C_1, D) is better than any solution lying between (C_1, D) and (C'', D'') as they can be expressed as convex combination of (C_1, D) and (C'', D'') and $\underline{\underline{Z}} < Z''$. According to theorem 2, for any solution in S, there exists a better solution on B_1. Hence (C_1, D) is atleast as

good as any neighbouring solution. Therefore (C_1, D) is a local optima. In linear programming, local optima is also a global optima. Hence the theorem follows.

Case (ii) is proved as follows:

$$Z' = w_1\alpha'_1 + r_2\beta'_2$$

$$\ddot{Z} = w_1\dot{\alpha}_1$$

Since $\bar{\alpha}_1 < \alpha'_1$, it follows that $\bar{Z} < Z'$ using theorem 2, $\dot{Z} < Z'$ and $\dot{Z} \leq \underline{Z}$, it can be proved that (C, D_1) is a local optima, which is also a global optima in linear programming. Hence (C, D_1) is an optimum solution.

Similar results can be proved for the cases when $(C, D_1) \varepsilon B_2 \cup B_3 \cup B_4$ and $(C_1, D) \varepsilon B_2 \cup B_3 \cup B_4$ lie between two adjacent extreme solutions then one of them is optimum depending on $\dot{Z} \geq \underline{Z}$ or $\bar{Z} < \underline{Z}$.

QED

We can infer from theorems (7) to (9) that the non-extreme solutions (C, D_1) and (C_1, D) behave as extreme solutions and hence the algorithm evaluates the objective function values of these non-extreme solutions.

If Z corresponding to (C, D) the current solution is greater than or equal to Z'' of (C'', D'') the previous extreme solution, then (C'', D'') is optimum and the algorithm stops. This is proved in the following theorem.

THEOREM 10: If $(C_1, D_1) \varepsilon R_1$ and $(C', D') \varepsilon B_1$, $(\bar{C}, \bar{D}) \varepsilon B_1$ and $(C'', D'') \varepsilon B_1$ be three adjacent extreme solutions satisfying $Z'' \geq \bar{Z}$ and $Z' \geq \bar{Z}$ then $(\bar{C}, \bar{D})$ is an optimum solution.

Proof: $(\bar{C}, \bar{D})$ is atleast as good as any other solution lying between $(\bar{C}, \bar{D})$ and (C', D') as they can be expressed as convex combination of these two solutions and $Z' \geq Z$. Similarly $(\bar{C}, \bar{D})$ is atleast as good as any other solution lying between $(\bar{C}, \bar{D})$ and (C'', D'') because they can be expressed as convex combination of these two solutions and $Z'' \geq \bar{Z}$. According to theorem 2, if $(C_1, D_1) \varepsilon R_1$ then a better solution exists on B_1 than any solution in S. Therefore $(\bar{C}, \bar{D})$ is a local optima, as it is better than any neighbouring solution. In linear programming local optima is also global optima and hence $(\bar{C}, \bar{D})$ is optimum. Similarly, this type of result can be proved for the cases, $(C_1, D_1) \varepsilon R_2 \cup R_3 \cup R_4$.

QED

If $D_1 \leq D_{min}$, then the algorithm moves along $B_1 \cup B_2$ and obtains an optimum solution. Otherwise the algorithm halts at $(C, D_1) \varepsilon B_1$ represented by J in Fig. 3.3. If $(C_1, D_1) \varepsilon R_1$ then (C, D_1) is an optimum solution which follows from the theorem 8. Otherwise it moves vertically up along JL in the direction of C_1.

We will discuss in detail how the algorithm increases the total cost C, while maintaining the total budget D constant. This is achieved by simultaneously augumenting the flow in two cycles.

An arc (i,j) is a forward arc if the flow is sent in the direction of the arc, and a backward arc if the flow is sent against the direction of the arc. We will refer to this as orientation of the arc. The basis structure of current solution is $(T \cup \{(p,q)\}, L, U)$ and its basis consists of exactly one cycle formed by the arc (p,q). This cycle is denoted by W_{pq}. Now, define a set

$$S^o = \{(i,j) \in L \cup U:\ \ddot{d}_{pq}/\dot{d}_{ij} < 0 \text{ and }$$

$$(\ddot{c}_{pq} - \frac{\ddot{d}_{pq}}{\dot{d}_{ij}} \times \ddot{c}_{ij} > 0)\}.$$

The above conditions are derived as follows.

<u>Criteria for Selecting a Non-Basic Arc for 2 Cycle Flow Augumentation:</u>

We will derive the conditions that the non-basic arc $(r, s) \in L \cup U$ must satisfy, such that by simultaneously augumenting flow in cycles W_{pq} and W_{rs}, the total cost increases and the total budget remains constant.

The basis structure is $(T \cup \{(p,q)\}, L, U)$.

Let Δ_{pq} be the flow to be circulated in cycle W_{pq}.
Let Δ_{rs} be the flow to be circulated in cycle W_{rs}.

For the total budget D to remain constant,

$$\Delta_{pq}\,\bar{d}_{pq} + \Delta_{rs}\,\bar{d}_{rs} = 0 \tag{3.13}$$

In order to increase the total cost C,

$$\Delta_{pq}\,\bar{c}_{pq} + \Delta_{rs}\,\bar{c}_{rs} > 0 \tag{3.14}$$

Using (3.13),

$$\frac{\Delta_{rs}}{\Delta_{pq}} = -\left(\frac{\bar{d}_{pq}}{\bar{d}_{rs}}\right) = K \tag{3.15}$$

Since $\Delta_{pq}, \Delta_{rs} > 0$, hence in order to satisfy (3.13), we get,

$$\frac{\bar{d}_{pq}}{\bar{d}_{rs}} < 0 \tag{3.16}$$

Dividing (3.14) by Δ_{pq} and substituting (3.15), we get,

$$\bar{c}_{pq} - \frac{\bar{d}_{pq}}{\bar{d}_{rs}} \times \bar{c}_{rs} > 0 \tag{3.17}$$

Hence if an arc (r, s) satisfies (3.16) and (3.17) and augumenting flow Δ_{pq} and $\Delta_{rs}K$ in cycles W_{pq} and W_{rs}, the total cost increases and total budget D remains constant.

If $S^o = \{\phi\}$ then reverse the orientation of the arc (p,q) and set $\bar{c}_{pq} = -\bar{c}_{pq}$, $\bar{d}_{pq} = -\bar{d}_{pq}$. Then, we again find S^o. If S^o

is not empty, select an arc $(r,s) \varepsilon S^o$ and enter it into the basis. This results in the formation of two cycles W_{pq} and W_{rs}. The orientation of a cycle W_{ij}, $(i,j) \varepsilon L \cup U$ is same as that of the arc (i,j). Let $\overline{W}_{ij}$ and $\underline{W}_{ij}$ be the sets of arcs, along and opposite to the orientation of the cycle W_{ij}, respectively. Let Δ_{pq} be the amount of flow to be augmented in W_{pq} and $K.\Delta_{pq}$ be the flow to be augmented in cycle W_{rs} where $K = - \bar{d}_{pq}/\bar{d}_{rs}$. Next, we determine Δ_{pq} as follows.

Consider an arc $(i,j) \varepsilon (W_{pq} \cap W_{rs})$. The flow in arc (i,j) after augmenting flow in W_{pq} and W_{rs} is,

$$\bar{x}_{ij} = x_{ij} + \Delta_{pq} I_1 + I_2 K \Delta_{pq}$$

$$= x_{ij} + \Delta_{pq} (I_1 + I_2 K)$$

where,

x_{ij} = existing flow in (i,j)

I_1 = + 1, if $(i,j) \varepsilon \overline{W}_{pq}$

I_1 = - 1, if $(i,j) \varepsilon \underline{W}_{pq}$

I_2 = + 1, if $(i,j) \varepsilon \overline{W}_{rs}$

I_2 = - 1, if $(i,j) \varepsilon \underline{W}_{rs}$

$\Delta_{pq} (I_1+I_2K)$ = resultant augmented flow in arc (i,j).

In order to satisfy the capacity constraints,

$$0 \leq x_{ij} + \Delta_{pq} (I_1+I_2K) \leq b_{ij}$$

If $(I_1 + I_2K) > 0$, then

$$\Delta_{pq} \leq \frac{(b_{ij} - x_{ij})}{(I_1+I_2K)}$$

If $(I_1 + I_2K) < 0$, then

$$\Delta_{pq} \leq - \frac{x_{ij}}{(I_1+I_2K)}$$

In this manner Δ_{pq} is calculated for all arcs $(i,j) \varepsilon (W_{pq} \cup W_{rs})$. Therefore,

$$\Delta_{pq} = \min_{(i,j)\varepsilon(W_{pq} \cup W_{rs})} \left\{- \frac{x_{ij}}{(I_1+I_2K)} \text{ if } (I_1+I_2K) < 0, \quad \frac{(b_{ij} - x_{ij})}{(I_1+I_2K)} \text{ if } I_1+I_2K > 0\right\}$$

$$\Delta_{pq} = \min. \left(\Delta_{pq}, \frac{(C_1-C)}{(\bar{c}_{pq} + K\bar{c}_{rs})}\right)$$

Once Δ_{pq} is obtained, the flows are updated in W_{pq} and W_{rs} cycles. One of the arcs $(u,v) \varepsilon (W_{pq} \cup W_{rs})$ leaves at one of its bounds respectively. The dual variables, $\bar{c}_{ij}$'s, $\bar{d}_{ij}$'s, $\forall (i,j) \varepsilon L \cup U$. C, D and Z are updated respectively.

Arc (p,q) for the next iteration is obtained as follows:

(a) If $(u,v) = (i,j) \varepsilon (W_{pq})$, set $(p,q) = (r,s)$.

(b) If $(u,v) = (i,j) \varepsilon (\{W_{pq} \cap W_{rs}\} \cup W_{rs})$, (p,q) does not change.

If the arc (p,q) is at one of its bounds, reverse the orientation of (p,q) and set $\bar{c}_{pq} = -\bar{c}_{pq}$, $\bar{d}_{pq} = -\bar{d}_{pq}$. By selecting an arc $(r,s) \in S^o$ at each iteration and augmenting flows in W_{pq}, and W_{rs}, the algorithm moves along J,L shown in Fig. 3.3. If $(C_1,D_1) \in S$, then the algorithm obtains this solution which is an optimum solution and hence stops. If S^o is empty then the current solution,

$$(C,\ D) \in B_2 \quad \text{if} \quad \frac{\bar{c}_{pq}}{\bar{d}_{pq}} > 0$$

or,

$$(C,\ D) \in B_3 \quad \text{if} \quad \frac{\bar{c}_{pq}}{\bar{d}_{pq}} \leq 0$$

according to the theorem 11. L in Fig. 3.3 represents this solution.

THEOREM 11: If $S^o = \{\phi\}$ then the current solution $(C,D) \in B_2$ if $\bar{c}_{pq}/\bar{d}_{pq} > 0$ and $(C,D) \in B_3$ if $\bar{c}_{pq}/\bar{d}_{pq} \leq 0$.

Proof: The necessary and sufficient conditions for $(C,\ D) \in B_2$ are, the non-basic arcs $(i,j) \in L \cup U$ whose $\bar{c}_{ij} > 0$ and $\bar{d}_{ij} \leq 0$, do not exist.

The necessary and sufficient conditions for $(C,\ D) \in B_3$ are, the non-basic arcs $(i,j) \in L \cup U$ whose $\bar{c}_{ij} > 0$ and $\bar{d}_{ij} \geq 0$, do not exist.

Suppose, $\dfrac{\bar{c}_{pq}}{\bar{d}_{pq}} > 0$

Arcs of the type $\dot{c}_{ij} \geq 0$ and $\dot{d}_{ij} < 0$ satisfy (3.16) and (3.17) if $\dot{c}_{pq} > 0$ and $\dot{d}_{pq} > 0$. Arcs of the type $\dot{c}_{ij} > 0$ and $d_{ij} = 0$ satisfy (3.16) and (3.17), if $c_{pq} < 0$ and $\dot{d}_{pq} < 0$.

Since $S^o = \{\emptyset\}$, the non-basic arcs (i,j) of the type $\dot{c}_{ij} \geq 0$ and $\dot{d}_{ij} \leq 0$ are absent. Therefore $(C, D) \in B_2$.

Suppose, $\dot{c}_{pq}/d_{pq} \leq 0$

Arcs of the type $c_{ij} \geq 0$ and $\dot{d}_{ij} > 0$ satisfy $\frac{\dot{d}_{pq}}{\dot{d}_{ij}} < 0$ and $c_{pq} - \frac{\dot{d}_{pq}}{\dot{d}_{ij}} \times \dot{c}_{ij} > 0$ if $\dot{c}_{pq} \geq 0$ and $\dot{d}_{pq} < 0$. Since $S^o = \{\emptyset\}$ the non-basic arcs (i,j) whose $c_{ij} > 0$ and $\dot{d}_{ij} \geq 0$, $\forall$ $\forall (i,j) \in L \cup U$ are absent. Hence $(C, D) \in B_3$.

QED

Now, the algorithm moves towards M along the curve B_2. According to theorem 2, if $(C_1,D_1) \in R_2$ the $(C^*, D^*) \in B_2$ and $D^* \geq D_1$, and $C^* \leq C_1$, it is justified in moving along L, M for finding an optimum solution.

Similarly if $(C_1, D_1) \in R_3$, the algorithm traverses along I, F, H, Y and finds an optimum solution. If $(C_1, D_1) \in R_3$ and $D_1 \geq D_{max}$, then the algorithm moves along B_4 and B_3 curves, and obtains an optimum solution.

3.6 Statement of Algorithm:

A formal statement of the algorithm organised in a manner suitable for computer implementation is given below:

Step 1: Set count to zero. Solve the minimum cost flow problem with $\sum_{(i,j)\varepsilon A} c_{ij}\, x_{ij}$ as the objective function value. Let (B, L, U) be the optimum basis structure obtained and X be the solution. Compute,

$$C \;=\; \sum_{(i,j)\varepsilon A} c_{ij}\, x_{ij}, \quad \text{and} \quad D \;=\; \sum_{(i,j)\varepsilon A} d_{ij}\, x_{ij}.$$

(C, D) is the minimum cost and budget respectively. Treating B as the basic tree, define the dual variables $\pi_j^{\,c}$'s and $\pi_j^{\,d}$'s as follows.

$$\pi_1^{\,c} \;=\; 0, \quad \pi_j^{\,c} = \pi_i^{\,c} + c_{ij}, \qquad \forall\ (i,j)\ \varepsilon\ B$$

$$\pi_1^{\,d} \;=\; 0, \quad \pi_j^{\,d} = \pi_i^{\,d} + d_{ij}, \qquad \forall\ (i,j)\ \varepsilon\ B$$

Define $\bar{c}_{ij}$ and $\bar{d}_{ij}$ for all non-basic arcs $(i,j)\ \varepsilon\ L \cup U$ as follows:

$$\bar{c}_{ij} \;=\; \begin{cases} \pi_i^{\,c} - \pi_j^{\,c} + c_{ij}\,, & \forall\ (i,j)\ \varepsilon\ L \\ \pi_j^{\,c} - \pi_i^{\,c} - c_{ij}\,, & \forall\ (i,j)\ \varepsilon\ U \end{cases}$$

$$\bar{d}_{ij} \;=\; \begin{cases} \pi_i^{\,d} - \pi_j^{\,d} + d_{ij}\,, & \forall\ (i,j)\ \varepsilon\ L \\ \pi_j^{\,d} - \pi_i^{\,d} - d_{ij}\,, & \forall\ (i,j)\ \varepsilon\ U \end{cases}$$

Compute, $\alpha_1 = (C - C_1)$

$\alpha_2 = (D - D_1)$

$\beta_1 = (C_1 - C)$

$\beta_2 = (D_1 - D)$

$$\alpha_i = \begin{cases} \alpha_i, & \text{if } \alpha_i > 0, \quad \forall\, i = 1,2 \\ 0, & \text{if } \alpha_i < 0, \end{cases}$$

$$\beta_i = \begin{cases} \beta_i, & \text{if } \beta_i > 0, \quad \forall\, i = 1,2 \\ 0, & \text{if } \beta_i < 0, \end{cases}$$

$$Z = w_1\alpha_1 + w_2\alpha_2 + r_1\beta_1 + r_2\beta_2$$

If $D_1 \leq D$ then flag $= -1$, else flag $= 1$.

Set $Z' = Z$, $X' = X$, $C' = C$, $D' = D$ and go to Step 2.

Step 2: Let $S' = \{(i,j) \in L \cup U: \bar{c}_{ij} \leq 0$ and flag $* \bar{d}_{ij} > 0\}$

If $S' = \phi$ then go to Step 3. Otherwise select an arc $(p,q) \in S'$ and enter it into the basis. Perform pivot iteration, up date x_{ij}'s.

If flag $* D >$ flag $* D_1$, then circulate Δ flow against the orientation of W_{pq} where $\Delta = \dfrac{D-D_1}{d_{pq}}$, and go to Step 7.

If $Z > Z'$ then X' is an optimum solution, Stop, otherwise, update (B,L,U), π_j^c's, π_j^d's, set $Z' = Z$, $X' = X$, $C' = C$ and $D' = D$. Repeat this step.

Step 3: Let $S' = \{(i,j) \in L \cup U: \bar{c}_{ij} > 0$ and $flag * \bar{d}_{ij} > 0\}$.
Define,

$$\mu_{ij} = \frac{\bar{c}_{ij}}{\bar{d}_{ij}} * flag.$$

If $S' = \{\phi\}$ then set flag $= -$ flag and go to Step 4.
Otherwise select an arc (p,q), $\mu_{pq} = \min_{(i,j)\in S'} (\mu_{ij})$, and enter it into the basis. Perform the pivot iteration, update x_{ij}'s.

If $C > C_1 > C'$ then circulate Δ flow against the orientation of W_{pq}, where $\Delta = (C-C_1)/\bar{c}_{pq}$. Set $Z' = Z$, $X' = X$, $C' = C$ and $D' = D$. Augment Δ flow in W_{pq} where $\Delta = (C-C_1)/\bar{c}_{pq}$.

If flag $* D >$ flag $* D_1$, then circulate Δ flow against the orientation of W_{pq} where $\Delta = (D-D_1)/\bar{d}_{pq}$, then check if $Z \geq Z'$ then X' is an optimum solution, stop. Otherwise, go to Step 7.

If $Z \geq Z'$ then X' is an optimum solution, stop.
Otherwise update (B,L,U), π_j^c's, π_j^d's, set $Z' = Z$, $X' = X$, $C' = C$, and $D' = D$. Repeat this step.

Step 4: Let $S' = \{(i,j) \in L \cup U: \bar{c}_{ij} > 0$ and $\bar{d}_{ij} = 0\}$.
If $S' = \phi$ then go to Step 5. Otherwise select an arc $(p,q) \in S'$ and enter it into the basis. Perform the pivot iteration, update x_{ij}'s.

If $C > C_1 > C'$ then circulate Δ flow against the orientation of W_{pq} where $\Delta = (C-C_1)/\bar{c}_{pq}$. X is an optimum solution, stop.

If $Z > Z'$ then X' is an optimum solution, stop. Otherwise, update (B,L,U), π_j^c's, π_j^d's, Set $Z' = Z$, $X' = X$, $C' = C$ and $D' = D$. Repeat this step.

Step 5. Let $S' = \{(i,j) \varepsilon L \cup U: \bar{c}_{ij} > 0 \text{ and flag} * \bar{d}_{ij} > 0\}$.
Define, $\mu_{ij} = \bar{c}_{ij} * \text{flag}/\bar{d}_{ij}$.
If $S' = \{\phi\}$ then X' is an optimum solution, stop.
Otherwise select an arc (p,q) $\mu_{pq} = \max\limits_{(i,j)\varepsilon S'} (\mu_{ij})$ and enter it into the basis, go to Step 6.

Step 6: Perform the pivot iteration, update x_{ij}'s.
If $C > C_1 > C'$ then circulate Δ flow against the orientation of W_{pq} where $\Delta = (C-C_1)/\bar{c}_{pq}$. Then if $Z \geq Z'$, X' is an optimum solution, stop. Otherwise X is an optimum solution, stop.

If $Z > Z'$ then X' is an optimum solution, stop. Otherwise update (B,L,U), π_j^c's, π_j^d's. Set $Z' = Z$, $X' = X$, $C' = C$ and $D' = D$. Repeat this step.

Step 7: If count = 2 then set $Z' = Z$, $X' = X$, $C' = C$, $D' = D$ and go to Step 8.

If $C \not> C_1$, then X is an optimum solution, stop. Otherwise the basis structure is $(T \cup \{(p,q)\}, L, U)$.

Define $S^o = \{(i,j) \in L \cup U: \dot{d}_{pq}/\dot{d}_{ij} < 0$ and

$(\dot{c}_{pq} - (\dot{d}_{pq}/d_{ij})*c_{ij} > 0)\}$. If S^o is empty then set count = count + 1, $\dot{c}_{pq} = -c_{pq}$, $\dot{d}_{pq} = -\dot{d}_{pq}$, status = -status and go to Step 7. Otherwise, select an arc $(r,s) \in S^o$ and enter it into the basis $(T \cup \{p,q\})$.

Two cycles are formed i.e. W_{pq}, W_{rs}. Define,

$K = -\dot{d}_{pq}/\dot{d}_{rs}$, set $T_{ij} = 0$, $\forall (i,j) \in A$.

Define,

$$T_{ij} = \begin{cases} T_{ij}+1, & \text{if } (i,j) \in \overline{W}_{pq} \\ T_{ij}-1, & \text{if } (i,j) \in W_{pq} \end{cases}$$

$$T_{ij} = \begin{cases} T_{ij}+K, & \text{if } (i,j) \in \overline{W}_{rs} \\ T_{ij}-K, & \text{if } (i,j) \in W_{rs} \end{cases}$$

$$\Delta = \min_{(i,j) \in (W_{pq} \cup W_{rs})} \left(-\frac{x_{ij}}{T_{ij}} \text{ if } T_{ij} < 0, \; \left(\frac{b_{ij}-x_{ij}}{T_{ij}}\right), \text{ if } T_{ij} > 0\right).$$

$$\Delta_{pq} = \min. \left(\Delta, \frac{(C_1 - C)}{(\dot{c}_{pq} + K * \dot{c}_{rs})}\right)$$

Update Δ_{pq} in W_{pq} and $K\,\Delta_{pq}$ in W_{rs}.

Define $S^o = \{(i,j) \in L \cup U: \dot{d}_{pq}/\dot{d}_{ij} < 0$ and $(\dot{c}_{pq} - (\dot{d}_{pq}/\dot{d}_{ij}) * \dot{c}_{ij} > 0)\}$. If S^o is empty then set count = count + 1, $\dot{c}_{pq} = -c_{pq}$, $\dot{d}_{pq} = -\dot{d}_{pq}$, status = –status and go to Step 7. Otherwise, select an arc $(r,s) \in S^o$ and enter it into the basis $(T \cup \{p,q\})$.

Two cycles are formed i.e. W_{pq}, W_{rs}. Define, $K = -\dot{d}_{pq}/\dot{d}_{rs}$, set $T_{ij} = 0$, $\forall (i,j) \in A$.

Define,

$$T_{ij} = \begin{cases} T_{ij}+1, & \text{if } (i,j) \in \overline{W}_{pq} \\ T_{ij}-1, & \text{if } (i,j) \in W_{pq} \end{cases}$$

$$T_{ij} = \begin{cases} T_{ij}+K, & \text{if } (i,j) \in \overline{W}_{rs} \\ T_{ij}-K, & \text{if } (i,j) \in W_{rs} \end{cases}$$

$$\Delta = \min_{(i,j)\in(W_{pq} \cup W_{rs})} \left(-\frac{x_{ij}}{T_{ij}} \text{ if } T_{ij} < 0, \quad \left(\frac{b_{ij}-x_{ij}}{T_{ij}}\right), \text{ if } T_{ij} > 0 \right).$$

$$\Delta_{pq} = \min.\left(\Delta, \frac{(C_1 - C)}{(\dot{c}_{pq} + K * \dot{c}_{rs})}\right)$$

Update Δ_{pq} in W_{pq} and $K\,\Delta_{pq}$ in W_{rs}.

Update (B,L,U), π_J^{c}'s, π_J^{d}'s.

The arc (p,q) for the next iteration is obtained as follows.

Let (u,v) be the leaving arc at its respective bound.

If $(u,v) = (i,j) \in W_{pq}$, set $(p,q) = (r,s)$

If $(u,v) = (i,j) \in (W_{pq} \cap W_{rs}) \cup W_{rs})$, (p,q) does not chang

If $x_{pq} = 0$ or $x_{pq} = b_{pq}$, then set $\bar{c}_{pq} = -\dot{c}_{pq}$, $\dot{d}_{pq} = -\dot{d}_{pq}$, status = – status. Repeat this step.

Step 8: The current basis structure is $(T \cup \{(p,q)\}, L, U)$.

If $\ddot{c}_{pq} = 0$ then the current solution X is optimum, stop.

If $c_{pq} < 0$, then set $c_{pq} = -\dot{c}_{pq}$, $\dot{d}_{pq} = -\tilde{d}_{pq}$, status = -status.

If $\dot{d}_{pq} > 0$, then set flag = 1 else set flag = -1,

Go to Step 6.

3.7 Numerical Example:

In this section a small-network flow problem is solved to illustrate the various steps of WGNF algorithm. The network is shown in Fig. 3.4. The numbers c_{ij}, b_{ij}, d_{ij} are indicated over each arc. An amount of 5 units of flow is sent from source 1 to sink 6. The aspiration levels and weights are specified as follows

$$w_1 = 0.15,\ w_2 = 0.10,\ r_1 = 0.35,\ r_2 = 0.4,\ C_1 = 165 \text{ and}$$
$$D_1 = 125.$$

The steps of the algorithm are summarized in Table 3.1. The ↑ indicates the basic pivot arc and ↓ indicates the basic arc leaving the basis. The graphs corresponding to the basis at various iterations are shown in Fig. 3.6. In these graphs, the basic pivot arc is drawn as dashed lines. The algorithm traced the path I, J, L, M as shown in Fig. 3.5 and obtained $C^* = 145$ and $D^* = 155$ as an optimum solution.

3.8 Mathematical Formulation of Interval Goal Network Flow Problem:

The mathematical formulation of interval goal network flow problem is as follows:

$$\text{Min. } Z = w_3\alpha_3 + w_4\alpha_4 + r_1\beta_1 + r_2\beta_2 \tag{3.18}$$

s.t.

$$\sum_{(j,i)\in I(i)} x_{ji} - \sum_{(i,j)\in O(i)} x_{ij} = \begin{cases} -V, & \text{if } i=1 \\ V, & \text{if } i=n, \ \forall\, i\in N \\ 0, & \text{otherwise} \end{cases} \tag{3.19}$$

$$0 \leq x_{ij} \leq b_{ij}, \quad \forall\, (i,j) \in A \tag{3.20}$$

$$\sum_{(i,j)\in A} c_{ij}\, x_{ij} + \beta_1 - \alpha_1 = C_1 \tag{3.21}$$

$$\sum_{(i,j)\in A} d_{ij}\, x_{ij} + \beta_2 - \alpha_2 = D_1 \tag{3.22}$$

$$\sum_{(i,j)\in A} c_{ij}\, x_{ij} + \beta_3 - \alpha_3 = C_2 \tag{3.23}$$

$$\sum_{(i,j)\in A} d_{ij}\, x_{ij} + \beta_4 - \alpha_4 = D_2 \tag{3.24}$$

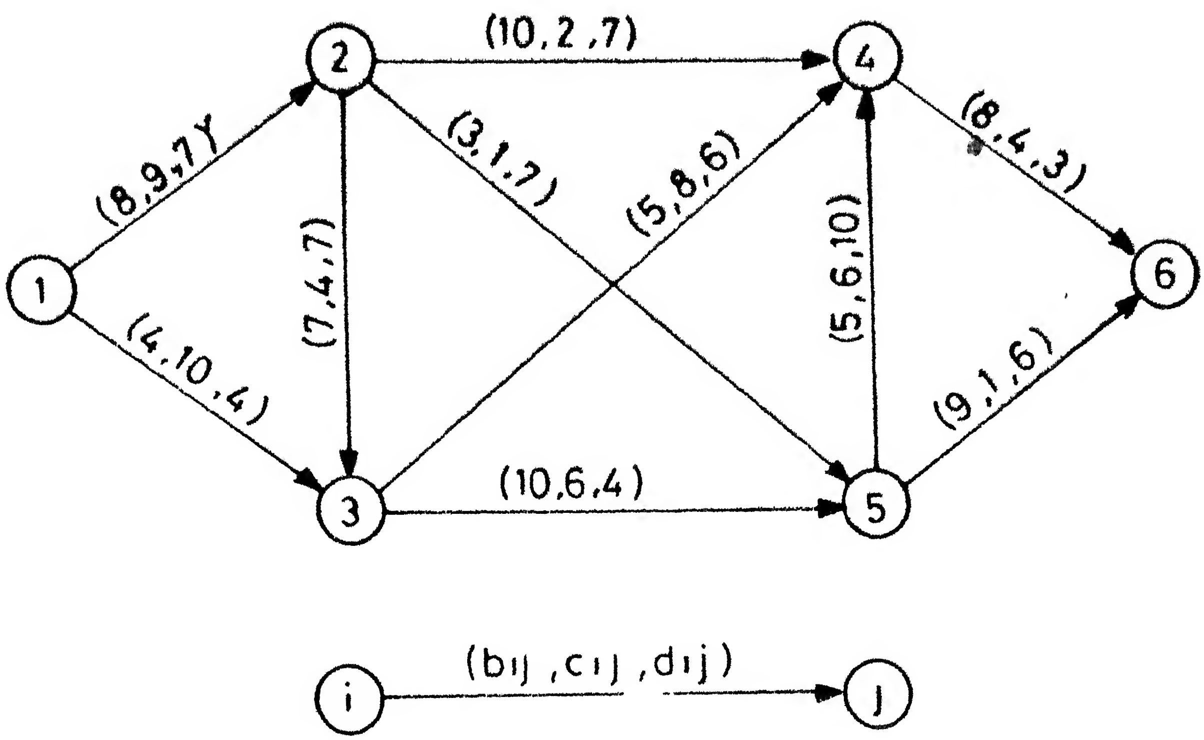

Fig. 3 4 Numerical example for the W(.F problem

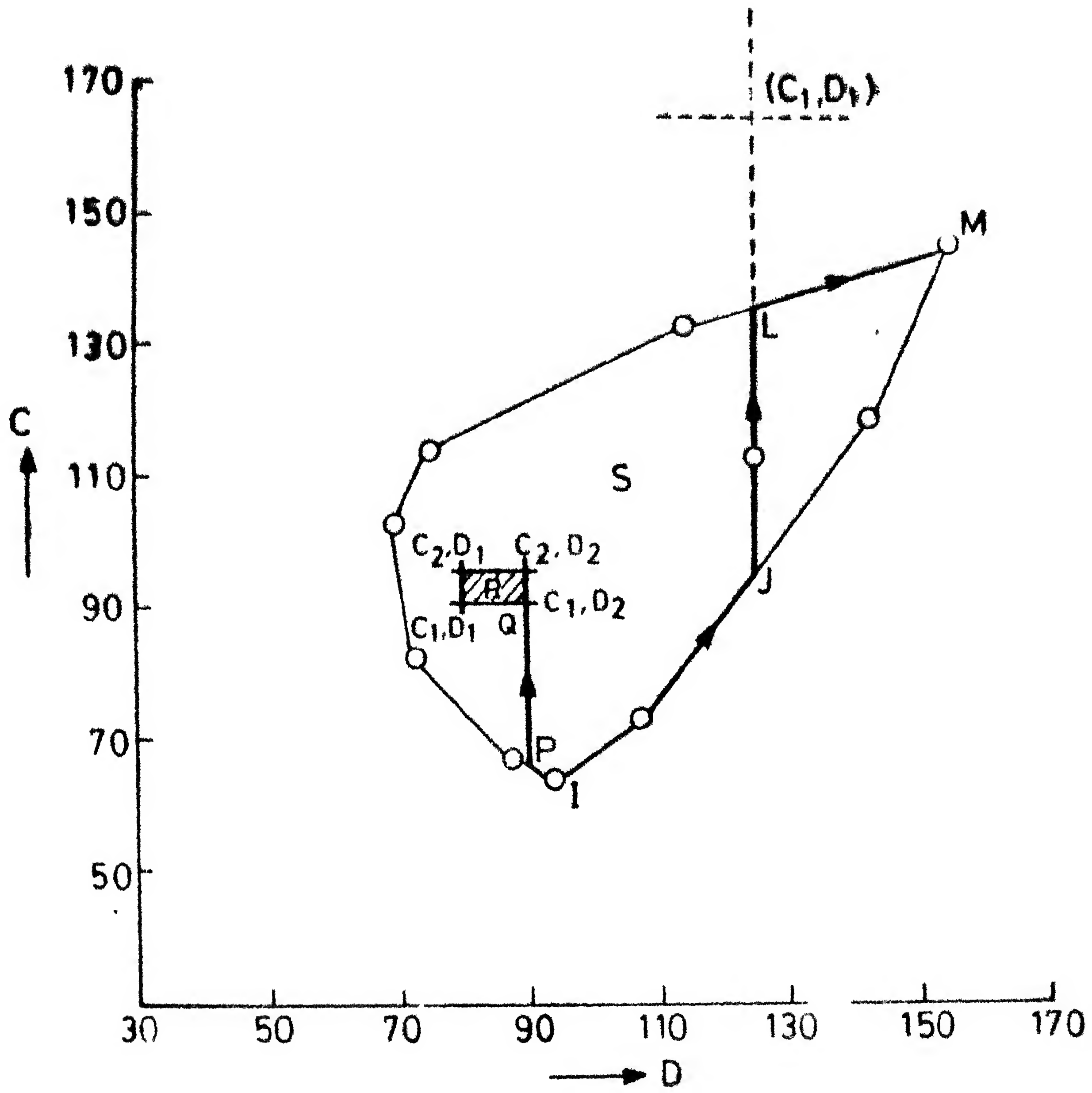

Fig. 3·5 Paths traced by WGNF algorithm and IGNF algorithm for the numerical example

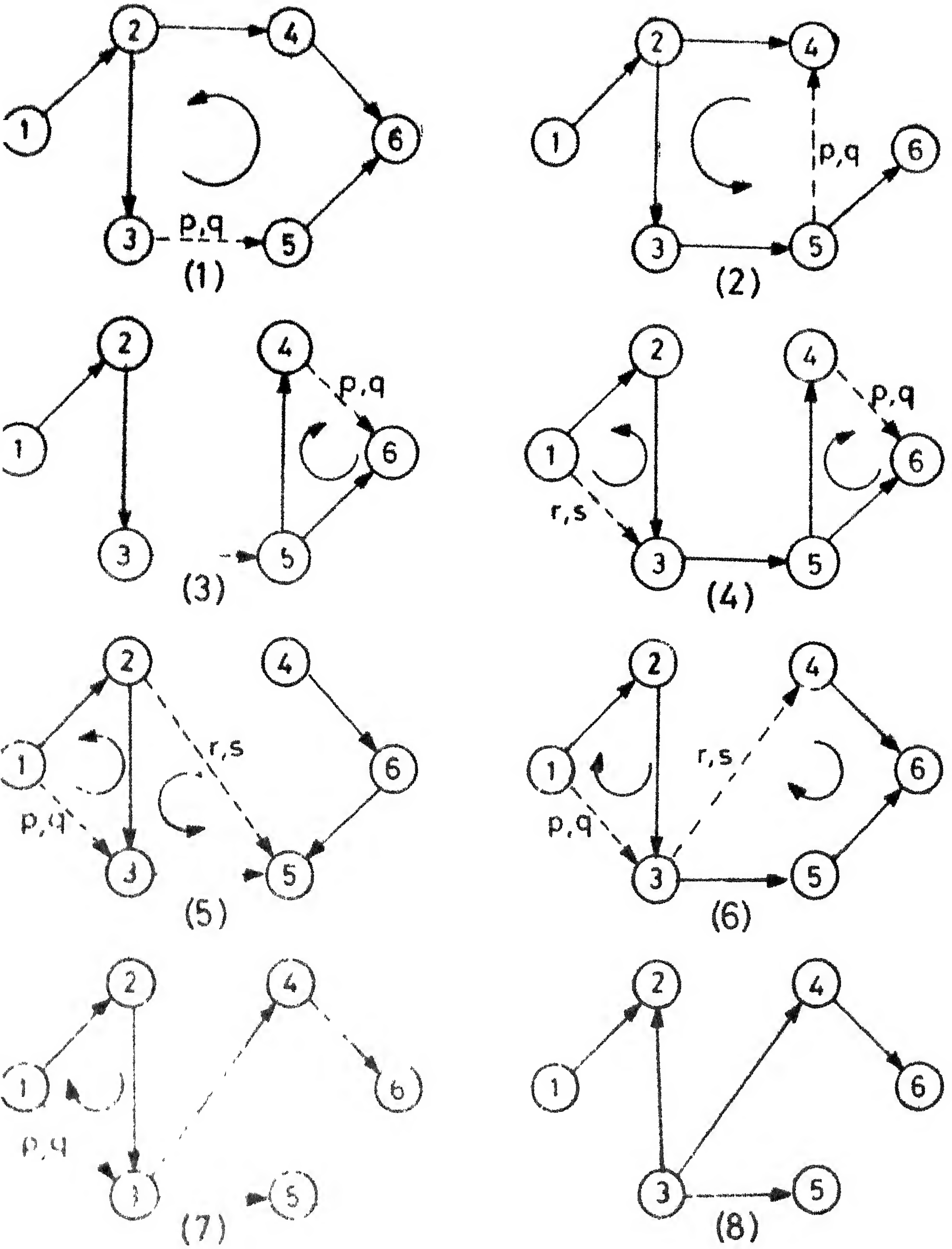

Fig 3.6 Basis in various interations

Table 3.1: Solution of the WGNF Problem.

	C	D	Z		Basic Tree Arcs					Non-tree Arcs				
				(i,j)	(1,2)	(2,4)	(2,3)	(4,6)	(5,6)	(1,3)	(2,5)	(3,4)	(3,5)	(5,4)
				x_{ij}	5	2	3	2	3	0	3	0	0	0
0	63	94	38.8	$\bar{c}_{ij}$	0	0	0	0	0	-3	4	10	5	9
				$\bar{d}_{ij}$	0	0	0	0 ↓	0	-10	-3	6	7 ↑(p,q)	7
				(i,j)	(1,2)	(2,4)	(2,3)	(3,5)	(5,6)	(1,3)	(2,5)	(3,4)	(4,6)	(5,4)
				x_{ij}	5	0	2	2	5	0	3	0	0	0
0.3	73	108	33.9	$\bar{c}_{ij}$	0	0	0	0	0	-3	9	10	-5	14
				$\bar{d}_{ij}$	0	0 ↓	0	0	0	-10	4	6	-7	14 ↑(p,q)
				(i,j)	(1,2)	(2,3)	(3,5)	(5,4)	(5,6)	(1,3)	(2,4)	(2,5)	(3,4)	(4,6)
				x_{ij}	5	2	2	0	5	0	0	3	0	0
0.72	73	108	33.9	$\bar{c}_{ij}$	0	0	0	0	0	-3	-14	9	10	-5
				$\bar{d}_{ij}$	0	0	0	0	0	-10	-14	4	6	-7 ↑(p,q)
				(i,j)	(1,2)	(2,3)	(3,5)	(5,4)	(5,6)	(1,3)	(2,4)	(2,5)	(3,4)	(4,6)
				x_{ij}	5	2	2	2.43	2.57	0	0	3	0	0
1.0	94.9	125	24.55	$\bar{c}_{ij}$	0	0	0	0	0	-3	-14	9	10	-5
				$\bar{d}_{ij}$	0	0	0	0 ↓	0	-10 ↑(r,s)	-14	4	6	-7 ↑(p,q)

				(i,j)	(1,2)	(2,3)	(4,6)	(3,5)	(5,6)	(1,3)	(2,4)	(2,5)	(3,4)	(5,4)
				x_{ij}	3.2	0.2	5.0	2.0	0	1.8	0	3	0	5
	112.6	125	18.34	$\bar{c}_{ij}$	0	0	0	0	0	−3	−5	9	5	−9
				$\bar{d}_{ij}$	0	0	0	0	0	−10 (p,q)	−7	4 (γ,δ)	−1	−1
				(i,j)	(1,2)	(2,3)	(3,5)	(4,6)	(5,6)	(1,3)	(2,4)	(2,5)	(3,4)	(5,4)
				x_{ij}	2	2	5	5	0	3	0	0	0	5
	136	125	10.15	$\bar{c}_{ij}$	0	0	0	00	0	−3	−5	−9	5	−9
				$\bar{d}_{ij}$	0	0	0	0	0	−10 (p,q)	−10	−4	−1 (γ,δ)	−1
				(i,j)	(1,2)	(2,3)	(3,4)	(3,5)	(4,6)	(1,3)	(2,4)	(2,5)	(5,4)	(5,6)
				x_{ij}	2	2	0	5	5	3	0	0	5	0
	136	125	10.15	$\bar{c}_{ij}$	0	0	0	0	0	3	−10	−9	−4	−5
				$\bar{d}_{ij}$	0	0	0	0	0	10 (p,q)	−6	−4	−8	−1
				(i,j)	(1,2)	(2,3)	(3,4)	(3,5)	(4,6)	(1,3)	(2,4)	(2,5)	(5,4)	(5,6)
				x_{ij}	5	5	0	5	5	0	0	0	5	0
).3	145	155	10.00	$\bar{c}_{ij}$	0	0	0	0	0	−3	−10	−9	−4	−5
				$\bar{d}_{ij}$	0	0	0	0	0	−10	−6	−4	−8	1

where,

C_1 = lower limit on total cost.

C_2 = upper limit on total cost.

D_1 = lower limit on total budget.

D_2 = upper limit on total budget.

We are interested in a solution between C_1 and C_2. Therefore β_1 and α_3 are minimized. Similarly if the solution is to be between D_1 and D_2, then β_2 and α_4 are minimized. By assigning proper weights, we obtain the objective function (3.18).

3.9 The Algorithm for Interval Goal Network Flow Problem:

The algorithm for interval goal network flow problem is similar to that of weighted goal network flow problem except for a few modifications.

The procedure to trace the path to reach an optimum solution is same as that of WGNF algorithm but the paths that these two algorithms take to obtain the optimum solutions are different. In interval goal problem, the decision maker specifies a range of aspiration levels for each of the goals and weighting factors for deviations from the ranges. The ranges specified form a rectangular region R in E whose four corner points are (C_1,D_1), (C_2,D_1), (C_2,D_2), (C_1,D_2) as shown in Fig. 3.7. The algorithm obtains an optimum solution $(X, \alpha_3, \alpha_4, \beta_1, \beta_2)$ by minimizing the sum of weighted deviations.

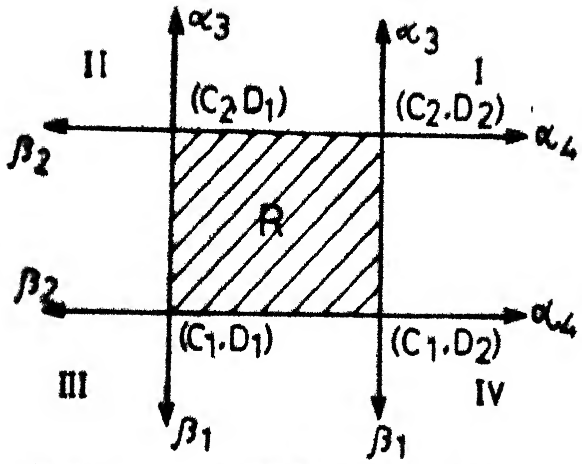

Fig. 3·7 Aspiration levels and devaiations for IGNF problem

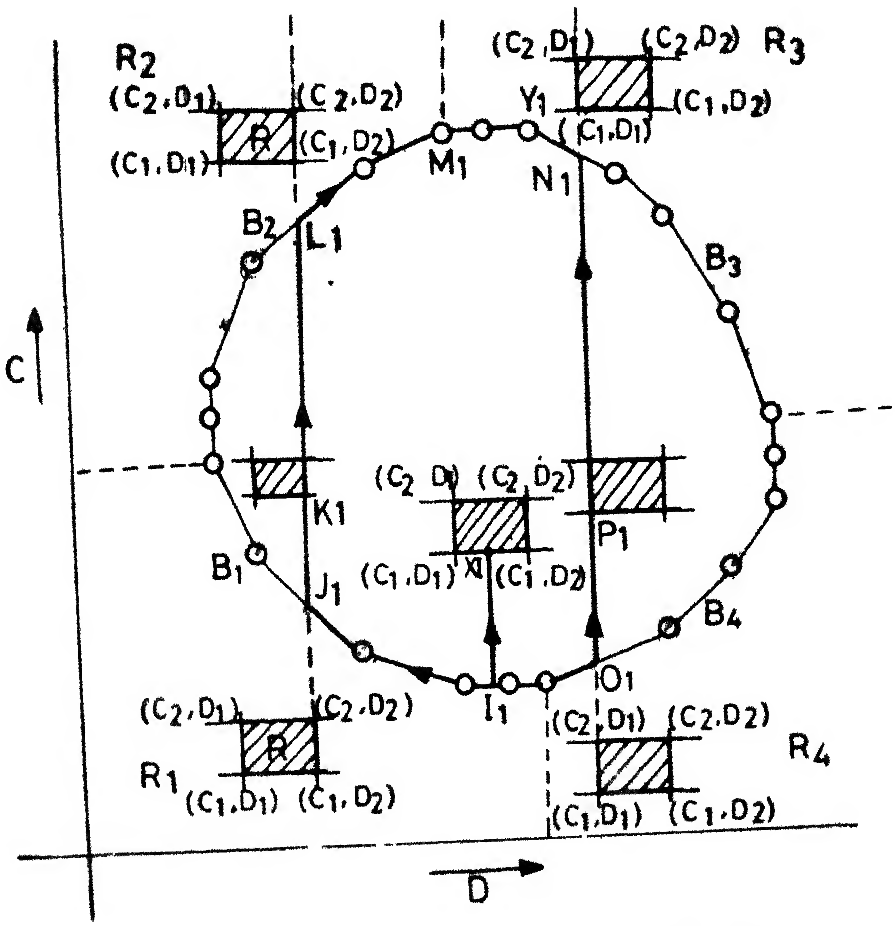

Fig 3 8 Various paths traced by IGNF alogorithm to obtain optimum solutions

From above, it can be inferred that for different positions of R, the IGNF algorithm considers one of the corner co-ordinates of R as aspiration levels and obtains an optimum solution like the WGNF algorithm. Therefore, all the theorems proved for WGNF algorithm are also valid for IGNF algorithm.

3.10 Numerical Example:

In this section a small network flow problem is solved to illustrates the various steps of IGNF algorithm. The network is shown in Fig. 3.4. The aspiration level ranges and the weighting factors specified for this problem are as follows:
$C_1 = 91$, $C_2 = 96$, $D_1 = 80$, $D_2 = 90$, $w_3 = 0.2$, $w_4 = 0.3$,
$r_1 = 0.15$, $r_2 = 0.35$.

The steps of the algorithm are summarized in Table 3.2. The $\uparrow$ indicates the basic pivot arc and $\downarrow$ indicates the basic arc leaving the basis. The algorithm traces I,P,Q path as shown in Fig. 3.5 and obtains C = 91 and D = 90 as an optimum solution. Q represents an optimum solution, in Fig. 3.5.

3.11 Computational Results:

The algorithms proposed in this chapter were coded, debugged and tested in Fortran - 10 on DEC-1090 multi-programming, time sharing computer system. A number of randomly generated network problems were solved with different number

Table 3.2: Solution of the IGNF Problem.

It.		C	D	Z		Basic Tree Arcs					Non-tree Arcs				
					(i,j)	(1,2)	(1,3)	(2,4)	(4,6)	(5,6)	(2,5)	(2,3)	(3,4)	(3,5)	(5,4)
					x_{ij}	5	0	2	3	2	3	0	0	0	0
1		63	94	5	$\bar{c}_{ij}$	0	0	0	0	0	4	3	7	2	9
					$\bar{d}_{ij}$	0	0	0	0	0	-3	10	-4	-3 ↑(p,q)	7
					(i,j)	(1,2)	(1,3)	(2,4)	(5,6)	(4,6)	(2,5)	(2,3)	(3,4)	(3,5)	(5,4)
					x_{ij}	3.67	1.33	0.67	4.33	0.67	3	0	0	4.33	0
2	0.67	65.7	90	3.8	$\bar{c}_{ij}$	0	0	0	0	0	4	1	-4	2	9
					$\bar{d}_{ij}$	0	0	0	0	0 ↓	-3	10 ↑(r,s)	3	-3 ↑(p,q)	-7
					(i,j)	(1,2)	(1,3)	(2,4)	(3,5)	(5,6)	(2,5)	(2,3)	(3,4)	(4,6)	(5,4)
					x_{ij}	3.2	1.8	0	2	5	3	0	0	0	0
3		67.6	90	3.51	$\bar{c}_{ij}$	0	0	0	0	0	6	1	7	-2	11
					$\bar{d}_{ij}$	0	0	0	0	0	↓ -6 ↑(r,s)	10 ↑(p,q)	-4	-2	11
					(i,j)	(1,2)	(1,3)	(2,4)	(3,5)	(5,6)	(2,5)	(2,3)	(3,4)	(4,6)	(5,4)
					x_{ij}	2	3	0	5	5	0	2	0	0	0
4		91	90	0	$\bar{c}_{ij}$	0	0	0	0	0	-6	1	7	-2	11
					$\bar{d}_{ij}$	0	0	0	0	0	6	10 ↑(p,q)	-4	3	4

of nodes and arcs and computational times were noted. A network generator was used for computational study which generates well-structured networks. The program first generates a skeleton network for a specified width and lengtl and then adds arcs randomly until the network contains a specified number of arcs.

Data structures based on augmented threaded index method[8] was implemented for storing the spanning tree. The tree was stored by means of thread indices, predecessor indices and the number of successors, each requiring an array of size n. Thread of a node i is a node that will be scanned in depth first search order of the sub-tree rooted at i. Predecessor of node i is the first node on the path from i to the root node of the tree. The number of successors of node i is the number of nodes in the subtree rooted at i. Link of a node i is the first arc on the path from i to the root node of the tree. If the arc is incident to i, it is stored as positive. If the arc is incident from i, it is stored as negative.

When an arc leaves the basis, it results in the formation of a hanging sub-tree. Thread indices help in scanning all the nodes of this sub-tree and thereby updating the dual variables. Predecessor indices are used for determining the minimum flow to be augmented, the leaving arc, and for updati the flows in the cycle formed, when a non-basic arc is added

to the basis. The number of successors for each node is maintained. It helps in determining the common node for the two paths when traversed from two ends of the entering arc towards the root node. Once the common node is known, it is easy to determine the minimum flow to be augmented in the cycle, leaving arc and updating the flow as well as the basis structure. Link is an array of size n that stores the arcs belonging to the basic tree. The storage for preserving the flows in arcs was reduced from m to n. This was achieved by storing the flows of basic arcs requiring an array of size n. The remaining non-basic arcs exist either at their lower or upper bounds, respectively. If the non-basic arc is at its upper bound, its capacity is made negative otherwise its capacity remains positive.

The main emphasis with computational results was laid on (i) to check the number of iterations performed by the algorithm and (ii) the computational time taken by the algorithms to get the optimum solutions. Problems sizes ranging from 10 nodes and 40 arcs to 200 nodes and 2500 arcs which includes both the sparse as well as dense networks were considered for computational study. Arc capacities were randomly generated between 5 to 100 whereas c_{ij}'s and d_{ij}'s were randomly generated between 1 to 50. Each problem was solved for three different aspiration levels and the computational times as well as the number of iterations are noted in Tables 3.3 and

3.4. About 20 problems of different sizes were solved for each of the algorithms. By comparing the columns (6), (8) and (10) of Table 3.3, we can infer that the time taken for solving $(C_1, D_1) \varepsilon R_2 \cup R_3$ is more than that for solving $(C_1, D_1) \varepsilon R_1 \cup R_4$ or $(C_1, D_1) \varepsilon S$. This is because, the optimum solution for $(C_1, D_1) \varepsilon R_2 \cup R_3$ lies on $B_2 \cup B_3$ and thereby traversing a longer path. No comparison can be made between (6) and (8) of Table 3.3 as the time taken to solve $(C_1, D_1) \varepsilon R_1 \cup R_4$ and $(C_1, D_1) \varepsilon S$ depends on the values of C_1 and D_1. One can infer similar results by comparing columns (6), (8) and (10) of Table 3.4 of IGNF problem. The reason being that WGNF algorithm and IGNF algorithm behave in a similar manner.

It is evident from the tables that both the algorithms can solve quite large problems in reasonable amount of time. For instance, problem of size 400 nodes and 2500 arcs was solved in about $2\frac{1}{2}$ minutes by both the algorithms. Infact the time taken by both the algorithms are comparable because IGNF algorithm is similar to WGNF algorithm except for a few modifications which will not alter computational times significantly.

Both the algorithms suggested by us are able to solve practically large problems in a reasonable amount of computer time. From the literature review we can conclude that no work was done in the field of goal programming techniques applied to

Table: 3.3: Computational Times of WGNF Algorithm.
(Execution Times in Seconds on DEC-1090 System)

Width	Length	Nodes	Arcs	$(C_1,D_1)\varepsilon R_1 U R_4$		$(C_1,D_1)\varepsilon S$		$(C_1,D_1)\varepsilon R_2 U R_3$	
				ITER	TIME	ITER	TIME	ITER	TIME
3	3	10	40	36	0.07	34	0.053	58	0.1
3	5	15	70	82	0.17	102	0.14	82	0.18
5	5	25	100	106	0.27	151	0.23	192	0.42
5	5	25	200	84	0.34	71	0.24	177	0.82
5	10	50	200	282	1.45	369	1.56	379	1.92
5	10	50	400	141	1.17	100	0.65	288	2.58
5	15	75	300	435	3.22	386	1.24	535	3.98
5	15	75	600	230	2.74	277	3.21	553	7.55
10	10	100	400	580	5.73	1115	4.51	778	7.53
10	10	100	1000	313	6.1	303	5.35	383	7.59

Table 3.3 continued

Width	Length	Nodes	Arcs	$(C_1,D_1)\varepsilon R_1 UR_4$ ITER	$(C_1,D_1)\varepsilon R_1 UR_4$ TIME	$(C_1,D_1)\varepsilon S$ ITER	$(C_1,D_1)\varepsilon S$ TIME	$(C_1,D_1)\varepsilon R_2 UR_3$ ITER	$(C_1,D_1)\varepsilon R_2 UR_3$ TIME
5	25	75	500	785	9.56	332	3.74	947	11.38
5	25	75	1200	1652	48.92	13370	99.64	3246	93.44
10	15	150	600	1062	15.42	1262	11.96	1369	19.59
10	15	150	1500	3226	118.45	4172	115.15	4334	155.5
7	25	175	700	1315	22.19	1711	23.28	1701	28.33
7	25	175	1600	3042	133.40	9480	147.19	4532	173.83
10	20	200	1000	403	8.02	672	8.93	915	20.40
10	20	200	2000	3674	146.39	9521	153.26	4756	181.57
20	20	400	1500	3709	139.37	9377	146.26	4375	173.69
20	20	400	2500	5213	172.56	11361	199.52	6857	208.18

Table 3.4: Computational Times of IGNF Algorithm.
(Execution Times in Seconds on DEC-1090 Systems)

Width	Length	Nodes	Arcs	$R \in R_1 \cup R_4$ ITER	$R \in R_1 \cup R_4$ TIME	$R \in S$ ITER	$R \in S$ TIME	$R \in R_2 \cup R_3$ ITER	$R \in R_2 \cup R_3$ TIME
3	3	10	40	36	0.07	34	0.053	58	0.11
3	5	15	70	82	0.17	102	0.139	82	0.18
5	5	25	100	114	0.32	124	0.22	167	0.47
5	5	25	200	88	0.39	136	0.36	191	0.95
5	10	50	200	243	1.26	541	1.86	391	2.03
5	10	50	400	158	1.37	610	1.91	379	2.71
5	15	75	300	397	2.93	302	1.14	535	3.98
5	15	75	600	237	2.89	280	3.07	602	8.40
10	10	100	400	607	6.09	410	2.16	791	7.78

Table 3.4 continued

Width	Length	Nodes	Arcs	$R \varepsilon R_1 \cup R_4$		$R \varepsilon S$		$R \varepsilon R_2 \cup R_3$	
				ITER	TIME	ITER	TIME	ITER	TIME
10	10	100	1000	313	6.06	458	5.52	1139	12.33
5	25	125	500	387	4.57	384	3.95	915	11.13
5	25	125	1200	1603	47.89	11053	85.37	3085	89.20
10	15	150	600	1096	16.05	2338	12.41	1374	19.72
10	15	150	1500	3361	123.31	11944	125.61	4444	159.60
7	25	175	700	1370	23.27	2695	28.91	1739	29.17
7	25	175	1600	3535	138.60	13692	186.14	4633	177.83
10	20	200	1000	426	8.648	1437	15.20	3507	26.31
10	20	200	2000	3572	141.35	9407	150.82	4947	183.58
20	20	400	1500	3657	138.07	9938	167.55	4489	179.56
20	20	400	2500	5008	165.09	10021	195.83	6517	203.18

network flow problems. This made us to explore this field, and suggest exact algorithms for weighted goal and interval goal network flow problems. One disadvantage with goal programming techniques is, the decision maker should specify proper weighting factors otherwise he may not obtain the preferred solution.

REFERENCES

1. Ahuja, R.K., Batra, J.L., and Gupta, S.K., The constrained minimum cost flow problem, Research paper, Industrial and Management Engineering Programme, Indian Institute of Technology, Kanpur (1980).

2. Ahuja, R.K., Batra, J.L., and Gupta, S.K., The constrained maximum flow problem, working paper, Industrial and Management Engineering Programme, Indian Institute of Technology, Kanpur (1980).

3. Ambrose, G., Hansen, D.R., and Lucien, D., Multi-objective Decision Analysis with Engineering and Business Applications, John Wiley and Sons, (1982).

4. Chen, S., and Saigal, R., A Primal Algorithm for Solving a Capacitated Network Flow Problem with Additional Linear Constraints, Networks 7 (1977), 59-80.

5. Dantzig, G.B., Linear Programming and Extensions, Princeton University Press, Princeton, N.J. (1963).

6. Glover, F., Network Applications in Industry and Government.

7. Ignizio, Jame, P., An Approach to the Modalling and Analysis of Multi-objective Generalised Networks, EJOR (Netherlands) 12 (1933) 4, 357-361.

8. Kennington, J.L., and Helgason, R.V., Algorithms for Networks Programming, John Wiley and Sons, Inc., New York, (1980).

9. Klingman, D., and Hultz, J., Solving Constrained Generalized Network Problems, Research Report CCS 257 Centre for Cybernetics Studies, University of Texas, Austin, Texas, Nov. 1976.

10. Klingman, D., and Mote, J., Solution Approaches for Network Flow Problems with Multiple Criteria, AMS (India) 1 (1932), Jan.

11. Lawler, E., Combinatorial Optimization, Networks and Matroids, Holt, Rinehart and Winston (1976).

12. Takashi, K., The Lexico-shortest Route Algorithm for Solving the Minimum Cost Flow Problem with an Additional Linear Constraint, Journal of Operations Research Society of Japan, Vol. 26, No. 3, Sept. 1983.

13. Zeleny, M., Multi-criteria Decision Making, McGraw-Hill Book Company, 1982.

```
*******************************************************************
* PROGRAM TO SOLVE BI-CRITERIA MINIMUM COST FLOW PROBLEM BY      *
* GOAL PROGRAMMING                                                *
*******************************************************************
          INTEGER WIDTH,SEED,ROOT,SOURCE,SINK,PREV,COUNT,FRONT,START,
     1            FINISH,ENTER,V,STATUS,O,OP,Z,X,R,
     2            TIME1,TIME2,TIME,CMIN,TR,
     3            PRED(405),THREAD(405),PI(405),POINT(405),PID(0:405),
     4            SCAN(405),HEAD(3000),CAP(3000),COST(3000),BUD(0:3000),
     5            PO,RS,TAILPO,HEADPO,TAILRS,HEADRS,CODE,CODE1,CODE2,OVER
          REAL MAX,MAX1,MAX2,MUE,MUEMAX,ICAP,IFLOW,MINIM,JFLOW
          DIMENSION NUMBER(405),LINK(405),LIST(405),FLOW(405),SUM(405)
          COMMON ITAIL,IHEAD,STATUS,DELTA,ICAP,LEG,ENTER,CMIN,DMIN,IW,
     1            LINK,FLOW,PRED,SUM,CAP,NUMBER,THREAD,PI,PID,IFLOW,K
     2            ,LEAVE,RFLOW
          TYPE 10
 10       FORMAT(1X,'>>',$)
          ACCEPT *,WIDTH,LENGTH,M,SEED,V,C1,D1
          CALL SETRAN(SEED)
          CALL NG4(WIDTH,LENGTH,N,M,SOURCE,SINK,POINT,HEAD)
          DO 20 J=1,M
          CAP(J)=IRAN(5,100)
          COST(J)=IRAN(1,50)
 20       BUD(J)=IRAN(1,50)
          READ(21,*)W1,W2,W3,W4
*******************************************************************
* THIS PART OF PROGRAM FORMS THE BASIS TREE AND CALCULATES THE   *
* DUAL VARIABLES.                                                 *
*******************************************************************
          CALL RTIME(TIME1)
          OVER=0
          S=1.0E-5
          FLAG=0
          RFLAG=0
          TFLOW=0
          DO 30 I=1,N
          PRED(I)=0
          PI(I)=0
          PID(I)=0
 30       CONTINUE
          LARGE=10000000
          C=0
          MINIM=1.0E+14
          BUD(0)=0
          ROOT=SOURCE
          MA=M+1
          CAP(MA)=LARGE
          COST(MA)=LARGE
          BUD(MA)=0.0
          PRED(SINK)=SOURCE
          LINK(SINK)=MA
          PREV=SINK;TOP=1;COUNT=0
          KOUNT=1
          SCAN(TOP)=ROOT
 40       IF(TOP.EQ.0) GO TO 70
          I=SCAN(TOP)
          IF(I.GT.0) GO TO 50
          I=-I
          NUMBER(I)=COUNT-LIST(I)
          TOP=TOP-1
          GO TO 40
 50       LIST(I)=COUNT
          COUNT=COUNT+1
          THREAD(PREV)=I
          PREV=I
          SCAN(TOP)=-I
          IF(KOUNT.GE.N)GO TO 40
          IX=POINT(I)
          IY=POINT(I+1)-1
          IF(IX.GT.IY) GO TO 40
          DO 60 J=IX,IY
          K=HEAD(J)
          IF(PRED(K).NE.0) GO TO 60
          PRED(K)=I
          LINK(K)=J
          TOP=TOP+1
          SCAN(TOP)=K
          KOUNT=KOUNT+1
 60       CONTINUE
          GO TO 40
 70       THREAD(PREV)=SINK
          THREAD(SINK)=SOURCE
          NUMBER(SINK)=1
          NUMBER(SOURCE)=N
 80       CONTINUE
          PI(1)=0
          PID(1)=0
          PID(0)=0
          PI(SINK)=LARGE
```

```
      PI(SINK)=LARGE
      PID(SINK)=0
      FLOW(1)=0
      FLOW(SINK)=V
      C=V*LARGE
      D=0.0
      I=SINK
   90 K=THREAD(I)
      PI(K)=PI(PRED(K))+COST(LINK(K))
      PID(K)=PID(PRED(K))+BUD(LINK(K))
      FLOW(K)=0
      IF(K.EQ.PREV)GO TO 100
      I=K
      GO TO 90
  100 CONTINUE
      CODE=5
  110 RSIGN=-1
      CMINIM=C
      DMINIM=D
*****************************************************************
* THIS PART OF THE PROGRAM SELECTS AN APPROPRIATE NON - BASIC   *
* ARC PQ FOR FLOW AUGMENTATION.                                 *
*****************************************************************
  120 SIGN=-1
      MUEMAX=LARGE
      IF(D1.GT.DMINIM)SIGN=1
  130 FLOWPQ=0
      ITER=ITER+1
      IF(D1.EQ.D.AND.C.GE.C1)GO TO 920
      IF(C1.EQ.C.AND.D.NE.D1.AND.(CODE.EQ.3.OR.CODE.EQ.4))GO TO 920
      Q=ABS(D-D1)
      IF((CODE.LE.2).AND.(Q.LE.S))GO TO 500
      ENTER=0
      DO 230 I=1,N
      START=POINT(I)
      FINISH=POINT(I+1)-1
      IF(START.GT.FINISH)GO TO 230
      DO 230 J=START,FINISH
      K=HEAD(J)
      IF(CAP(J).LT.0)GO TO 140
      CBAR=COST(J)+PI(I)-PI(K)
      DBAR=BUD(J)+PID(I)-PID(K)
      STATUS=1
      GO TO 150
  140 CBAR=PI(K)-PI(I)-COST(J)
      DBAR=PID(K)-PID(I)-BUD(J)
      STATUS=-1
  150 GO TO(160,210,170,210,200)CODE
  160 IF(CBAR.GT.0.OR.(SIGN*DBAR).GE.0)GO TO 230
      GO TO 180
  170 IF(CBAR.LE.0.OR.DBAR.NE.0)GO TO 230
  180 ENTER=J
      CMIN=CBAR
      DMIN=DBAR
      STATPQ=STATUS
      IHEAD=K
      ITAIL=I
      PQ=J
  190 GO TO 270
  200 IF(CBAR.GE.0.OR.MUEMAX.LE.CBAR)GO TO 230
      MUEMAX=CBAR
      GO TO 220
  210 IF(CBAR.LE.0.OR.(SIGN*DBAR).GE.0)GO TO 230
      MUE=(RSIGN*CBAR)/(SIGN*DBAR)
      IF(MUE.GE.MUEMAX)GO TO 230
      MUEMAX=MUE
  220 ENTER=J
      CMIN=CBAR
      DMIN=DBAR
      STATPQ=STATUS
      PQ=J
      IHEAD=K
      ITAIL=I
  230 CONTINUE
      IF(CODE.EQ.5.AND.ENTER.NE.0)GO TO 270
      IF(CODE.EQ.5.AND.ENTER.EQ.0)CODE=1
      IF(ENTER.NE.0)GO TO 260
      IF(CODE.EQ.4.AND.ENTER.EQ.0)GO TO 920
      CODE=CODE+1
      IF(CODE.NE.4)GO TO 120
  240 SIGN=-SIGN
      RSIGN=-RSIGN
  250 MUEMAX=0
      GO TO 130
  260 MAX1=(C1-C)/CMIN
      MAX2=(D1-D)/DMIN
      MAX=MAX1
      IF((MAX1.GT.MAX2.AND.MAX2.GT.0).OR.MAX1.LE.0)MAX=MAX2
```

```
**********************************************************************
* THIS PART OF THE PROGRAM DETERMINES THE LEAVING ARC AND THE        *
* AMOUNT OF FLOW TO BE AUGMENTED.                                    *
**********************************************************************
  270     ICAP=CAP(ENTER)
          LEG=0
          FLOWPQ=IABS(CAP(PQ))
          IF(STATPQ.EQ.1)FLOWPQ=0
          IF(FLAG.EQ.0)GO TO 280
          FLOWPQ=TFLOW
          FLAG=0
  280     IF(ICAP.LT.0)GO TO 290
          STATUS=1
          IX=ITAIL
          IY=IHEAD
          DELTA1=ICAP-FLOWPQ
          GO TO 300
  290     ICAP=-ICAP
          CMIN=-CMIN
          DMIN=-DMIN
          STATUS=-1
          IX=IHEAD
          IY=ITAIL
          DELTA1=FLOWPQ
  300     DELTA=DELTA1
          RFLOW=FLOWPQ
  310     IF(IX.EQ.IY)GO TO 390
          IF(NUMBER(IX).GT.NUMBER(IY))GO TO 350
          J=LINK(IX)
          IF(J.LT.0)GO TO 320
          CHANGE=CAP(J)-FLOW(IX)
          GO TO 330
  320     CHANGE=FLOW(IX)
  330     IF(CHANGE.GE.DELTA)GO TO 340
          DELTA=CHANGE
          K=IX
          LEG=1
  340     IX=PRED(IX)
          GO TO 310
  350     J=LINK(IY)
          IF(J.LT.0)GO TO 360
          CHANGE=FLOW(IY)
          GO TO 370
  360     CHANGE=CAP(-J)-FLOW(IY)
  370     IF(CHANGE.GE.DELTA)GO TO 380
          DELTA=CHANGE
          K=IY
          LEG=-1
  380     IY=PRED(IY)
          GO TO 310
  390     CONTINUE
          IW=IX
          DELTA1=DELTA
          IF(CODE.NE.5)GO TO 410
          STATUS=STATPQ
          ENTER=PQ
          C=C+CMIN*STATUS*DELTA
          D=D+DMIN*STATUS*DELTA
          IDIR=-1
          IF(D1.GT.D)IDIR=1
          IF(DELTA1.LT.ICAP)GO TO 400
          CALL AUGMEN
          GO TO 110
  400     CALL UPDATE
          GO TO 110
  410     IF(DELTA1.LE.MAX)GO TO 460
          DELTA=MAX
          ADD=CMIN*STATUS*DELTA
          ADD1=DMIN*STATUS*DELTA
          C=C+ADD
          D=D+ADD1
          CALL UPDAT1(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,IDIR,DELTA,RFLAG)
          C=C-ADD
          D=D-ADD1
          IF(OVER.EQ.1)GO TO 920
          C=C+ADD
          D=D+ADD1
          IF(ABS(D-D1).LE.S.AND.CODE.GE.3)GO TO 430
          IF(D.NE.D1.AND.CODE.LE.2)GO TO 430
          IF(ABS(C-C1).GT.S.AND.CODE.GE.3)GO TO 430
  420     CALL AUGMEN
          CAP(ENTER)=-CAP(ENTER)
          GO TO 130
  430     DELTA=MAX1
          IF(MAX.EQ.MAX1)DELTA=MAX2
          IF(DELTA.LT.0.OR.DELTA.GE.DELTA1)GO TO 450
          DELTA=DELTA-MAX
          ADD=CMIN*STATUS*DELTA
```

```
      PI(ST,K)=LARGE
      ADD2=DMIN*STATUS*DELTA
      C=C+ADD1
      D=D+ADD2
      IF(ABS(C-C1).LE.S.AND.CODE.GE.3)GO TO 420
      CALL UPDATI(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,IDIR,DELTA,RFLAG)
      IF(OVER.EQ.1)GO TO 440
      DELTA=DELTA+MAX
      CALL AUGMEN
      GO TO 920
440   DELTA=MAX
      C=C-ADD1
      D=D-ADD2
      CALL AUGMEN
      GO TO 920
450   DELTA=DELTA1-MAX
460   ADD=CMIN*STATUS*DELTA
      ADD1=DMIN*STATUS*DELTA
      C=C+ADD
      D=D+ADD1
      DELTA=DELTA1
      IF(DELTA1.LE.S)GO TO 480
470   CALL UPDATI(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,IDIR,DELTA,RFLAG)
      C=C-ADD
      D=D-ADD1
      IF(OVER.EQ.1.AND.(DELTA.EQ.(DELTA1-MAX)))GO TO 440
      IF(OVER.EQ.1)GO TO 920
      C=C+ADD
      D=D+ADD1
480   IF(LEG.EQ.0)CALL AUGMEN
      IF(LEG.NE.0)CALL UPDATE
490   IF(CODE.EQ.4)GO TO 250
      GO TO 120
***********************************************************************
* THIS PART OF THE PROGRAM SELECTS THE NON - BASIC ARC RS FOR         *
* SIMULTANEOUS FLOW AUGMENTATION IN TWO CYCLES.                       *
***********************************************************************
500   CBPQ=CMIN
      DBPQ=DMIN
      TAILPQ=ITAIL
      HEADPQ=IHEAD
      FLOWPQ=DELTA
      IF(STATPQ.EQ.-1)FLOWPQ=-CAP(PQ)-DELTA
510   CODE2=1
520   RS=0
      ITER=ITER+1
      DO 530 I=1,N
530   SUM(I)=0.0
      IF(ABS(C-C1).LE.S)GO TO 920
      DO 560 I=1,N
      START=POINT(I)
      FINISH=POINT(I+1)-1
      IF(START.GT.FINISH)GO TO 560
      DO 560 J=START,FINISH
      K=HEAD(J)
      IF(CAP(J).LT.0)GO TO 540
      CBAR=COST(J)+PI(I)-PI(K)
      DBAR=BUD(J)+PID(I)-PID(K)
      STATUS=1
      GO TO 550
540   CBAR=PI(K)-PI(I)-COST(J)
      DBAR=PID(K)-PID(I)-BUD(J)
      STATUS=-1
550   IF(DBAR.EQ.0)GO TO 560
      IF(DBAR/DBPQ.GE.0)GO TO 560
      IF((CBPQ-(DBPQ*CBAR)/DBAR).LE.0)GO TO 560
      RS=J
      TAILRS=I
      HEADRS=K
      STATRS=STATUS
      RATIO=-DBPQ/DBAR
      CBRS=CBAR
      DBRS=DBAR
      FLOWRS=0.0
      IF(STATRS.EQ.-1)FLOWRS=-CAP(RS)
      GO TO 590
560   CONTINUE
      IF(CODE2.EQ.2.AND.RS.EQ.0)GO TO 570
      CBPQ=-CBPQ
      DBPQ=-DBPQ
      CAP(PQ)=-CAP(PQ)
      STATPQ=-STATPQ
      CODE2=CODE2+1
      GO TO 520
570   CODE=4
      CALL UPDATI(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,IDIR,DELTA,RFLAG)
      IF(CBPQ.GT.0)GO TO 580
      CBPQ=-CBPQ
      DBPQ=-DBPQ
      CAP(PQ)=-CAP(PQ)
```

```
      PI(STEK)=LARGE
  580 IF(DBPQ.LT.0.AND.IDIR.EQ.-1)RFLAG=-1
      IF(DBPQ.GT.0.AND.IDIR.EQ.1)RFLAG=+1
      IF(RFLAG.NE.0)SIGN=-SIGN
      FLAG=1
      TFLOW=FLOWPQ
      GO TO 240
*****************************************************************
* THIS PART OF THE PROGRAM DETERMINES THE LEAVING ARC AND THE   *
* MINIMUM FLOW TO BE AUGMENTED IN TWO CYCLES SIMULTANEOUSLY     *
*****************************************************************
  590 CODE1=1
      DELTA1=FLOWPQ
      IF(STATPQ.EQ.1)DELTA1=CAP(PQ)-FLOWPQ
      LEG=0
      DELTA2=IABS(CAP(RS))/RATIO
      IF(DELTA2.LT.DELTA1)DELTA1=DELTA2
  600 ENTER=PQ
      STATUS=STATPQ
      RIN=1
      INC=1
      ITAIL=TAILPQ
      IHEAD=HEADPQ
  610 IF(STATUS.EQ.-1)GO TO 620
      IX=ITAIL
      IY=IHEAD
      GO TO 630
  620 IX=IHEAD
      IY=ITAIL
  630 IF(IX.EQ.IY)GO TO 790
      IF(NUMBER(IX).GT.NUMBER(IY))GO TO 710
      IF(CODE1.EQ.3)GO TO 660
      J=LINK(IX)
      IF(J.LT.0)GO TO 640
      SUM(IX)=SUM(IX)+RIN
      GO TO 650
  640 SUM(IX)=SUM(IX)-RIN
  650 IF(CODE1.EQ.1)GO TO 700
  660 IF(SUM(IX))670,700,680
  670 DELTA2=-FLOW(IX)/SUM(IX)
      GO TO 690
  680 J=IABS(LINK(IX))
      DELTA2=(CAP(J)-FLOW(IX))/SUM(IX)
  690 IF(DELTA2.GE.DELTA1) GO TO 700
      DELTA1=DELTA2
      LEG=INC
      K1=IX
      TOTAL=FLOW(K1)+DELTA1*SUM(K1)
  700 IX=PRED(IX)
      GO TO 630
  710 IF(CODE1.GT.2)GO TO 740
      J=LINK(IY)
      IF(J.LT.0)GO TO 720
      SUM(IY)=SUM(IY)-RIN
      GO TO 730
  720 SUM(IY)=SUM(IY)+RIN
  730 IF(CODE1.EQ.1)GO TO 780
  740 IF(SUM(IY))750,780,760
  750 DELTA2=-FLOW(IY)/SUM(IY)
      GO TO 770
  760 J=IABS(LINK(IY))
      DELTA2=(CAP(J)-FLOW(IY))/SUM(IY)
  770 IF(DELTA2.GE.DELTA1)GO TO 780
      DELTA1=DELTA2
      LEG=-INC
      K1=IY
      TOTAL=FLOW(K1)+DELTA1*SUM(K1)
  780 IY=PRED(IY)
      GO TO 630
  790 IF(CODE1.EQ.1)IW1=IX
      IF(CODE1.EQ.2)IW2=IX
      CODE1=CODE1+1
      IF(CODE1.GT.3)GO TO 800
      IF(CODE1.EQ.3)GO TO 600
      ENTER=RS
      ITAIL=TAILRS
      IHEAD=HEADRS
      STATUS=STATRS
      RIN=RATIO
      INC=2
      GO TO 610
*****************************************************************
* THIS PART OF THE PROGRAM UPDATES THE FLOW IN BOTH THE CYCLES  *
* BY CALLING AN APPROPRIATE UPDATE SUBROUTINE AND DETERMINES    *
* THE ARC PQ.                                                   *
*****************************************************************
  800 DELTA2=(C1-C)/(CBPQ+RATIO*CBRS)
      IF(DELTA2.GT.0.AND.DELTA2.LT.DELTA1)DELTA1=DELTA2
      C=C+DELTA1*CBPQ+RATIO*DELTA1*CBRS
      D=D+DELTA1*DBPQ+RATIO*DELTA1*DBRS
```

```
      PI(SINK)=LARGE
      IF(IABS(LEG).GT.0)GO TO 840
      DELTA=DELTA1*RATIO
      ITAIL=TAILRS
      IHEAD=HEADRS
      STATUS=STATRS
      FLOWRS=FLOWRS+STATRS*DELTA
      IW=IW2
      ENTER=RS
      CALL AUGMEN
      DELTA=DELTA1
      ITAIL=TAILPQ
      IHEAD=HEADPQ
      STATUS=STATPQ
      FLOWPQ=FLOWPQ+STATPQ*DELTA
      IW=IW1
      ENTER=PQ
      CALL AUGMEN
      IF(FLOWRS.NE.0.AND.FLOWRS.NE.IABS(CAP(RS)))GO TO 820
C     IF(ABS(FLOWRS).GE.S.AND.ABS(FLOWRS-IABS(CAP(RS))))GO TO 830
      STATPQ=-STATPQ
      CBPQ=-CBPQ
      DBPQ=-DBPQ
  810 IF(FLOWPQ.LE.S.OR.ABS(FLOWPQ-IABS(CAP(PQ))).LE.S)GO TO 510
      STATPQ=-STATPQ
      CAP(PQ)=-CAP(PQ)
      CBPQ=-CBPQ
      DBPQ=-DBPQ
      GO TO 510
  820 FLOWPQ=FLOWRS
      CBPQ=-CBRS
      DBPQ=-DBRS
      PQ=RS
      TAILPQ=TAILRS
      HEADPQ=HEADRS
      STATPQ=-STATRS
  830 IF(FLOWPQ.LE.0.OR.ABS(FLOWPQ-IABS(CAP(PQ))).LE.S)GO TO 510
      CAP(PQ)=-CAP(PQ)
      STATPQ=-STATPQ
      CBPQ=-CBPQ
      DBPQ=-DBPQ
      GO TO 510
  840 IF(IABS(LEG).EQ.2)GO TO 880
      ITAIL=TAILRS
      IHEAD=HEADRS
      STATUS=STATRS
      DELTA=DELTA1*RATIO
      FLOWRS=FLOWRS+STATRS*DELTA
      IW=IW2
      ENTER=RS
      CALL AUGMEN
      ITAIL=TAILPQ
      IHEAD=HEADPQ
      STATUS=STATPQ
      DELTA=DELTA1
      IW=IW1
      ENTER=PQ
      CMIN=CBPQ
      DMIN=DBPQ
      IF(STATUS.EQ.1)GO TO 850
      CMIN=-CMIN
      DMIN=-DMIN
  850 ICAP=IABS(CAP(PQ))
      K=K1
      RFLOW=FLOWPQ
      CALL UPDATE
      CAP(LEAVE)=-IABS(CAP(LEAVE))
      IF(TOTAL.LE.S)CAP(LEAVE)=-CAP(LEAVE)
      FLOWPQ=FLOWRS
      PQ=RS
      TAILPQ=TAILRS
      HEADPQ=HEADRS
      CBPQ=COST(PQ)+PI(TAILPQ)-PI(HEADPQ)
      DBPQ=BUD(PQ)+PID(TAILPQ)-PID(HEADPQ)
      IF(STATRS.EQ.1)GO TO 860
      CBPQ=-CBPQ
      DBPQ=-DBPQ
  860 STATPQ=-STATRS
      CBPQ=-CBPQ
      DBPQ=-DBPQ
  870 GO TO 830
  880 ENTER=PQ
      ITAIL=TAILPQ
      IHEAD=HEADPQ
      STATUS=STATPQ
      DELTA=DELTA1
      IW=IW1
      FLOWPQ=FLOWPQ+STATPQ*DELTA1
      CALL AUGMEN
      ITAIL=TAILRS
```

```
      PI(SINK)=LARGE
      IHEAD=HEADRS
      STATUS=STATRS
      DELTA=DELTA1*RATIO
      IW=IW2
      LEG=LEG/2
      ENTER=RS
      K=K1
      CMIN=CBRS
      DMIN=DBRS
      IF(STATUS.EQ.1)GO TO 890
      CMIN=-CMIN
      DMIN=-DMIN
  890 ICAP=IABS(CAP(RS))
      REFLOW=FLOWRS
      CALL UPDATE
      CAP(LEAVE)=-IABS(CAP(LEAVE))
      IF(TOTAL.LE.S)CAP(LEAVE)=-CAP(LEAVE)
      CBPQ=COST(PQ)+PI(TAILPQ)-PI(HEADPQ)
      DBPQ=BUD(PQ)+PID(TAILPQ)-PID(HEADPQ)
      IF(STATPQ.EQ.1)GO TO 900
      CBPQ=-CBPQ
      DBPQ=-DBPQ
  900 STATPQ=-STATPQ
      CBPQ=-CBPQ
      DBPQ=-DBPQ
  910 GO TO 810
  920 CALL RTIME(TIME2)
      WRITE(22,930)C,D,TIME,ITER
  930 FORMAT(1X,'COST=',F14.4,' BUDGET=',F14.4,' TIME=',I8,'ITER=',I8)
      STOP
      END
```

```
******************************************************************
* THIS SUBROUTINE UPDATES THE OBJECTIVE FUNCTION OF FIXED        *
* WEIGHTED GOAL PROGRAMMING.                                     *
******************************************************************
      SUBROUTINE UPDAT1(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,IDIR,DELTA,
     1      RFLAG)
      REAL MIN,MINIM
      INTEGER OVER
      IF(DELTA.GE.(1.0E-6))GO TO 80
      IF(RFLAG)30,5,10
5     IF(IDIR)10,80,30
10    IF(C.GT.C1)GO TO 40
      MIN=W3*(C1-C)+W2*(D-D1)
70    OVER=1
      IF(MIN.GE.MINIM)GO TO 80
      MINIM=MIN
      OVER=0
80    RETURN
40    MIN=W1*(C-C1)+W2*(D-D1)
      GO TO 70
30    IF(C.GT.C1)GO TO 50
      MIN=W3*(C1-C)+W4*(D1-D)
      GO TO 70
50    MIN=W1*(C-C1)+W4*(D1-D)
      GO TO 70
      END
```

```
      PI(SINK)=LARGE
*****************************************************************
* THIS PROGRAM OBTAINS AN OPTIMUM SOLUTION FOR INTERVAL GOAL   *
* PROGRAMMING APPLIED TO BI-CRITERIA NETWORK FLOW PROBLEM.     *
*****************************************************************
      INTEGER WIDTH,SEED,ROOT,SOURCE,SINK,PREV,COUNT,FRONT,START,
     1        FINISH,ENTER,V,STATUS,Q,QP,Z,X,R,
     2        TIME1,TIME2,TIME,CMIN,TR,
     3        PRED(405),THREAD(405),PI(405),POINT(405),PID(0:405),
     4        SCAN(405),HEAD(3000),CAP(3000),COST(3000),BUD(0:3000),
     5        PQ,RS,TAILPQ,HEADPQ,TAILRS,HEADRS,CODE,CODE1,CODE2,OVER
      REAL MAX,MAX1,MAX2,MUE,MUEMAX,ICAP,IFLOW,MINIM,JFLOW
      DIMENSION NUMBER(405),LINK(405),LIST(405),FLOW(405),SUM(405)
      COMMON ITAIL,IHEAD,STATUS,DELTA,ICAP,LEG,ENTER,CMIN,DMIN,IW,
     1        LINK,FLOW,PRED,SUM,CAP,NUMBER,THREAD,PI,PID,IFLOW,K
     2        ,LEAVE,RFLOW
      TYPE 10
   10 FORMAT(1X,'>>'$)
      ACCEPT *,WIDTH,LENGTH,M,SEED,V,C1,D1,C2,D2
      CALL SETRAN(SEED)
      CALL NG4(WIDTH,LENGTH,N,M,SOURCE,SINK,POINT,HEAD)
      DO 20 J=1,M
      CAP(J)=IRAN(5,100)
      COST(J)=IRAN(1,50)
   20 BUD(J)=IRAN(1,50)
      READ(21,*)W1,W2,W3,W4
      CALL RTIME(TIME1)
******************************************************************
* THIS PART OF THE PROGRAM FORMS THE BASIS TREE ALONG WITH THE
* PREDECESSOR,THREAD INDICES, NO. OF SUCCESSORS AND CALCULATES   *
* THE DUAL VARIABLES PI'S AND PID'S.                             *
******************************************************************
      OVER=0
      FLAG=0
      S=1.0E-5
      RFLAG=0
      TFLOW=0
      DO 30 I=1,N
      PRED(I)=0
      PI(I)=0
      PID(I)=0
   30 CONTINUE
      LARGE=1000000
      C=0
      MINIM=1.0E+14
      BUD(0)=0
      ROOT=SOURCE
      MA=M+1
      CAP(MA)=LARGE
      COST(MA)=LARGE
      BUD(MA)=0.0
      PRED(SINK)=SOURCE
      LINK(SINK)=MA
      PREV=SINK;TOP=1;COUNT=0
      KOUNT=1
      SCAN(TOP)=ROOT
   40 IF(TOP.EQ.0) GO TO 70
      I=SCAN(TOP)
      IF(I.GT.0) GO TO 50
      I=-I
      NUMBER(I)=COUNT-LIST(I)
      TOP=TOP-1
      GO TO 40
   50 LIST(I)=COUNT
      COUNT=COUNT+1
      THREAD(PREV)=I
      PREV=I
      SCAN(TOP)=-I
      IF(KOUNT.GE.N)GO TO 40
      IX=POINT(I)
      IY=POINT(I+1)-1
      IF(IX.GT.IY) GO TO 40
      DO 60 J=IX,IY
      K=HEAD(J)
      IF(PRED(K).NE.0) GO TO 60
      PRED(K)=I
      LINK(K)=J
      TOP=TOP+1
      SCAN(TOP)=K
      KOUNT=KOUNT+1
   60 CONTINUE
      GO TO 40
   70 THREAD(PREV)=SINK
      THREAD(SINK)=SOURCE
      NUMBER(SINK)=1
      NUMBER(SOURCE)=N
   80 CONTINUE
      PI(I)=0
      PID(I)=0
      P[illegible](I)=0
```

```
      PI(SINK)=LARGE
      PID(SINK)=0
      FLOW(I)=0
      FLOW(SINK)=V
      C=V*LARGE
      D=0.0
      I=SINK
   90 K=THREAD(I)
      PI(K)=PI(PRED(K))+COST(LINK(K))
      PID(K)=PID(PRED(K))+BUD(LINK(K))
      FLOW(K)=0
      IF(K.EQ.PREV)GO TO 100
      I=K
      GO TO 90
  100 CONTINUE
      CODE=5
  110 RSIGN=-1
      CMINIM=C
      DMINIM=D
  120 SIGN=-1
*********************************************************************
* THIS PART OF THE PROGRAM SELECTS AN APPROPRIATE NON-BASIC ARC PO *
* FROM A SET OF NON-BASIC ARCS FOR FLOW AUGUMENTATION.              *
*********************************************************************
      MUEMAX=LARGE
      IF(D2.LT.DMINIM)SIGN=1
  130 FLOWPO=0
      IF(DMINIM.GE.D1.AND.DMINIM.LE.D2.AND.C.GE.C1.AND.CODE.NE.5)GO TO
     1      880
      IF((ABS(D-D1).LE.S.OR.ABS(D2-D).LE.S).AND.CODE.LE.2)GO TO 460
      IF(ABS(C-C1).LE.S.AND.(CODE.EQ.3.OR.CODE.EQ.4))GO TO 880
      IF((ABS(D-D1).LE.S.OR.ABS(D-D2).LE.S).AND.(C.GE.C1)
     1       .AND.(CODE.LE.2))GO TO 880
      IF(D.GE.D1.AND.D.LE.D2.AND.C.GE.C1.AND.C.LE.C2)GO TO 880
      ENTER=0
      ITER=ITER+1
      DO 230 I=1,N
      START=POINT(I)
      FINISH=POINT(I+1)-1
      IF(START.GT.FINISH)GO TO 230
      DO 230 J=START,FINISH
      K=HEAD(J)
      IF(CAP(J).LT.0)GO TO 140
      CBAR=COST(J)+PI(I)-PI(K)
      DBAR=BUD(J)+PID(I)-PID(K)
      STATUS=1
      GO TO 150
  140 CBAR=PI(K)-PI(I)-COST(J)
      DBAR=PID(K)-PID(I)-BUD(J)
      STATUS=-1
  150 GO TO(160,210,170,210,200)CODE
  160 IF(CBAR.GT.0.OR.(SIGN*DBAR).GE.0)GO TO 230
      GO TO 180
  170 IF(CBAR.LE.0.OR.DBAR.NE.0)GO TO 230
  180 ENTER=J
      CMIN=CBAR
      DMIN=DBAR
      STATPO=STATUS
      THEAD=K
      ITAIL=I
      PO=J
  190 GO TO 270
  200 IF(CBAR.GE.0.OR.MUEMAX.LE.CBAR)GO TO 230
      MUEMAX=CBAR
      GO TO 220
  210 IF(CBAR.LE.0.OR.(SIGN*DBAR).GE.0)GO TO 230
      MUE=(RSIGN*CBAR)/(SIGN*DBAR)
      IF(MUE.GE.MUEMAX)GO TO 230
      MUEMAX=MUE
  220 ENTER=J
      CMIN=CBAR
      DMIN=DBAR
      STATPO=STATUS
      PO=J
      THEAD=K
      ITAIL=I
  230 CONTINUE
      IF(CODE.EQ.5.AND.ENTER.NE.0)GO TO 270
      IF(CODE.EQ.5.AND.ENTER.EQ.0)CODE=1
      IF(ENTER.NE.0)GO TO 260
      IF(CODE.EQ.4.AND.ENTER.EQ.0)GO TO 880
      CODE=CODE+1
      IF(C1.LE.C.AND.C.LE.C2.AND.CODE.EQ.3)GO TO 880
      IF(CODE.NE.4)GO TO 120
  240 SIGN=-SIGN
      RSIGN=-RSIGN
  250 MUEMAX=0
      GO TO 130
  260 MAXI=(C2-C)/CMIN
```

```
      IF(CODE.GT.3)MAX1=(C1-C)/CMIN
      MAX2=(D1-D)/DMIN
      IF(D2.LT.DMINIM)MAX2=(D2-D)/DMIN
      MAX=MAX1
      IF((MAX1.GT.MAX2.AND.MAX2.GT.0).OR.MAX1.LT.0)MAX=MAX2
*************************************************************************
* THIS PART OF THE PROGRAM DETERMINS THE LEAVING ARC FROM THE CYCLE *
* AND THE MINIMUM FLOW TO BE AUGUMENTED IN THE CYCLE.                 *
*************************************************************************
  270 ICAP=CAP(ENTER)
      LEG=0
      FLOWPQ=IABS(CAP(PQ))
      IF(STATPQ.EQ.1)FLOWPQ=0
      IF(FLAG.EQ.0)GO TO 280
      FLOWPQ=IFLOW
      FLAG=0
  280 IF(ICAP.LT.0)GO TO 290
      STATUS=1
      IX=ITAIL
      IY=IHEAD
      DELTA1=ICAP-FLOWPQ
      GO TO 300
  290 ICAP=-ICAP
      CMIN=-CMIN
      DMIN=-DMIN
      STATUS=-1
      IX=IHEAD
      IY=ITAIL
      DELTA1=FLOWPQ
  300 DELTA=DELTA1
      RFLOW=FLOWPQ
  310 IF(IX.EQ.IY)GO TO 390
      IF(NUMBER(IX).GT.NUMBER(IY))GO TO 350
      J=LINK(IX)
      IF(J.LT.0)GO TO 320
      CHANGE=CAP(J)-FLOW(IX)
      GO TO 330
  320 CHANGE=FLOW(IX)
  330 IF(CHANGE.GE.DELTA)GO TO 340
      DELTA=CHANGE
      K=IX
      LEG=1
  340 IX=PRED(IX)
      GO TO 310
  350 J=LINK(IY)
      IF(J.LT.0)GO TO 360
      CHANGE=FLOW(IY)
      GO TO 370
  360 CHANGE=CAP(-J)-FLOW(IY)
  370 IF(CHANGE.GE.DELTA)GO TO 380
      DELTA=CHANGE
      K=IY
      LEG=-1
  380 IY=PRED(IY)
      GO TO 310
  390 CONTINUE
      IW=IX
      DELTA1=DELTA
      IF(CODE.NE.5)GO TO 410
      STATUS=STATPQ
      ENTER=PQ
      C=C+CMIN*STATUS*DELTA
      D=D+DMIN*STATUS*DELTA
      DIR=-1
      IF(D1.GT.D)DIR=1
      IF(DELTA1.LT.ICAP)GO TO 400
      CALL AUGMEN
      GO TO 110
  400 CALL UPDATE
      GO TO 110
  410 IF(DELTA1.LE.MAX)GO TO 450
      DELTA=MAX
      ADD=CMIN*STATUS*DELTA
      ADD1=DMIN*STATUS*DELTA
      C=C+ADD
      D=D+ADD1
      CALL UPDAT1(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,DIR,DELTA,RFLAG
     1           ,C2,D2)
      C=C-ADD
      D=D-ADD1
      IF(OVER.EQ.1)GO TO 880
      C=C+ADD
      D=D+ADD1
      IF((ABS(D-D1).LE.S.OR.ABS(D-D2).LE.S).AND.CODE.GE.3)GO TO 420
      IF(ABS(D-D1).GE.S.AND.ABS(D-D2).GE.S)GO TO 420
      CALL AUGMEN
      CAP(ENTER)=-CAP(ENTER)
      GO TO 110
  420 DELTA=MAX1
```

```
      IF(MAX.EQ.MAX1)DELTA=MAX2
      IF(DELTA.LT.0.OR.DELTA.GE.DELTA1)GO TO 440
      DELTA=DELTA-MAX
      ADD1=CMIN*STATUS*DELTA
      ADD2=DMIN*STATUS*DELTA
      C=C+ADD1
      D=D+ADD2
      CALL UPDAT1(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,DIR,DELTA,RFLAG
     1      ,C2,D2)
      IF(OVER.EQ.1)GO TO 430
      DELTA=DELTA+MAX
      CALL AUGMEN
      GO TO 880
430   DELTA=MAX
      C=C-ADD1
      D=D-ADD2
      CALL AUGMEN
      GO TO 880
440   DELTA=DELTA1-MAX
450   ADD=CMIN*STATUS*DELTA
      ADD1=DMIN*STATUS*DELTA
      C=C+ADD
      D=D+ADD1
      DELTA=DELTA1
      CALL UPDAT1(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,DIR,DELTA,RFLAG
     1      ,C2,D2)
      C=C-ADD
      D=D-ADD1
      IF(OVER.EQ.1.AND.(DELTA.EQ.(DELTA1-MAX)))GO TO 430
      IF(OVER.EQ.1)GO TO 880
      C=C+ADD
      D=D+ADD1
      IF(LEG.EQ.0)CALL AUGMEN
      IF(LEG.NE.0)CALL UPDATE
      IF(CODE.EQ.4)GO TO 250
      GO TO 120
460   CBPQ=CMIN
      DBPQ=DMIN
      TAILPQ=ITAIL
      HEADPQ=IHEAD
      FLOWPQ=DELTA
      IF(STATPQ.EQ.-1)FLOWPQ=-CAP(PQ)-DELTA
***********************************************************************X
* THIS PART OF THE PROGRAM SELECTS AN ARC RS FROM A SET OF              *
* NON-BASIC ARCS FOR FLOW AUGUMENTATION IN TWO CYCLES SIMULTANEOUSLY*
***********************************************************************
470   CODE2=1
480   RS=0
      DO 490 I=1,N
490   SUM(I)=0.0
      IF(C.GE.C1.AND.C.LE.C2)GO TO 880
      ITER=ITER+1
      DO 520 I=1,N
      START=POINT(I)
      FINISH=POINT(I+1)-1
      IF(START.GT.FINISH)GO TO 520
      DO 520 J=START,FINISH
      K=HEAD(J)
      IF(CAP(J).LT.0)GO TO 500
      CBAR=COST(J)+PI(I)-PI(K)
      DBAR=BUD(J)+PID(I)-PID(K)
      STATUS=1
      GO TO 510
500   CBAR=PI(K)-PI(I)-COST(J)
      DBAR=PID(K)-PID(I)-BUD(J)
      STATUS=-1
510   IF(DBAR.EQ.0)GO TO 520
      IF(DBAR/DBPQ.GE.0)GO TO 520
      IF((CBPQ-(DBPQ*CBAR)/DBAR).LE.0)GO TO 520
      RS=J
      TAILRS=I
      HEADRS=K
      STATRS=STATUS
      RATIO=-DBPQ/DBAR
      CBRS=CBAR
      DBRS=DBAR
      FLOWRS=0.0
      IF(STATRS.EQ.-1)FLOWRS=-CAP(RS)
      GO TO 550
520   CONTINUE
      IF(CODE2.EQ.2.AND.RS.EQ.0)GO TO 530
      CBPQ=-CBPQ
      DBPQ=-DBPQ
      CAP(PQ)=-CAP(PQ)
      STATPQ=-STATPQ
      CODE2=CODE2+1
      GO TO [illegible]
530   CODE=[illegible]
      CALL UPDAT1(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,DIR,DELTA,RFLAG
```

```
      IF(CBPQ.GT.0)GO TO 540
      CBPQ=-CBPQ
      DBPQ=-DBPQ
      CAP(PQ)=-CAP(PQ)
540   IF(DBPQ.LT.0.AND.DIR.EQ.-1)RFLAG=-1
      IF(DBPQ.GT.0.AND.DIR.EQ.1)RFLAG=+1
      IF(RFLAG.NE.0)SIGN=-SIGN
      FLAG=1
      TFLOW=FLOWPQ
      GO TO 240
************************************************************************
* THIS PART OF THE PROGRAM DETERMINES THE LEAVING ARC AND THE          *
* MININMUM FLOW TO BE AUGUMENTED IN THE TWO CYCLES SIMULTANEOUSLY*
************************************************************************
550   CODE1=1
      DELTA1=FLOWPQ
      IF(STATPQ.EQ.1)DELTA1=CAP(PQ)-FLOWPQ
      LEG=0
      DELTA2=IABS(CAP(RS))/RATIO
      IF(DELTA2.LT.DELTA1)DELTA1=DELTA2
560   ENTER=PQ
      STATUS=STATPQ
      RIN=1
      INC=1
      ITAIL=TAILPQ
      IHEAD=HEADPQ
570   IF(STATUS.EQ.-1)GO TO 580
      IX=ITAIL
      IY=IHEAD
      GO TO 590
580   IX=IHEAD
      IY=ITAIL
590   IF(IX.EQ.IY)GO TO 750
      IF(NUMBER(IX).GT.NUMBER(IY))GO TO 670
      IF(CODE1.EQ.3)GO TO 620
      J=LINK(IX)
      IF(J.LT.0)GO TO 600
      SUM(IX)=SUM(IX)+RIN
      GO TO 610
600   SUM(IX)=SUM(IX)-RIN
610   IF(CODE1.EQ.1)GO TO 660
620   IF(SUM(IX))630,660,640
630   DELTA2=-FLOW(IX)/SUM(IX)
      GO TO 650
640   J=IABS(LINK(IX))
      DELTA2=(CAP(J)-FLOW(IX))/SUM(IX)
650   IF(DELTA2.GE.DELTA1) GO TO 660
      DELTA1=DELTA2
      LEG=INC
      K1=IX
      TOTAL=FLOW(K1)+DELTA1*SUM(K1)
660   IX=PRED(IX)
      GO TO 590
670   IF(CODE1.GT.2)GO TO 700
      J=LINK(IY)
      IF(J.LT.0)GO TO 680
      SUM(IY)=SUM(IY)-RIN
      GO TO 690
680   SUM(IY)=SUM(IY)+RIN
690   IF(CODE1.EQ.1)GO TO 740
700   IF(SUM(IY))710,740,720
710   DELTA2=-FLOW(IY)/SUM(IY)
      GO TO 730
720   J=IABS(LINK(IY))
      DELTA2=(CAP(J)-FLOW(IY))/SUM(IY)
730   IF(DELTA2.GE.DELTA1)GO TO 740
      DELTA1=DELTA2
      LEG=-INC
      K1=IY
      TOTAL=FLOW(K1)+DELTA1*SUM(K1)
740   IY=PRED(IY)
      GO TO 590
750   IF(CODE1.EQ.1)IW1=IX
      IF(CODE1.EQ.2)IW2=IX
      CODE1=CODE1+1
      IF(CODE1.GT.3)GO TO 760
      IF(CODE1.EQ.3)GO TO 560
      ENTER=RS
      ITAIL=TAILRS
      IHEAD=HEADRS
      STATUS=STATRS
      RIN=RATIO
      INC=2
      GO TO 570
760   DELTA2=(C1-C)/(CBPQ+RATIO*CBRS)
      IF(DELTA2.GT.0.AND.DELTA2.LT.DELTA1)DELTA1=DELTA2
      C=C+DELTA1*CBPQ+RATIO*DELTA1*CBRS
      D=D+DELTA1*DBPQ+RATIO*DELTA1*DBRS
```

```
**********************************************************************
* THIS PART OF THE PROGRAM CALLS AN APPROPRIATE UPDATE SUBROUTINE    *
* FOR UPDATING FLOW,INDICES AND DUAL VARIABLES AND FINDS THE ARC PQ *
**********************************************************************
      IF(IABS(LEG).GT.0)GO TO 800
      DELTA=DELTA1*RATIO
      ITAIL=TAILRS
      IHEAD=HEADRS
      STATUS=STATRS
      FLOWRS=FLOWRS+STATRS*DELTA
      IW=IW2
      ENTER=RS
      CALL AUGMEN
      DELTA=DELTA1
      ITAIL=TAILPQ
      IHEAD=HEADPQ
      STATUS=STATPQ
      FLOWPQ=FLOWPQ+STATPQ*DELTA
      IW=IW1
      ENTER=PQ
      CALL AUGMEN
      IF(FLOWRS.NE.0.AND.FLOWRS.NE.IABS(CAP(RS)))GO TO 780
      STATPQ=-STATPQ
      CBPQ=-CBPQ
      DBPQ=-DBPQ
  770 IF(FLOWPQ.LE.S.OR.ABS(FLOWPQ-IABS(CAP(PQ))).LE.S)GO TO 470
      STATPQ=-STATPQ
      CAP(PQ)=-CAP(PQ)
      CBPQ=-CBPQ
      DBPQ=-DBPQ
      GO TO 470
  780 FLOWPQ=FLOWRS
      CBPQ=-CBRS
      DBPQ=-DBRS
      PQ=RS
      TAILPQ=TAILRS
      HEADPQ=HEADRS
      STATPQ=-STATRS
  790 IF(FLOWPQ.LE.S.OR.ABS(FLOWPQ-IABS(CAP(PQ))).LE.S)GO TO 470
      CAP(PQ)=-CAP(PQ)
      STATPQ=-STATPQ
      CBPQ=-CBPQ
      DBPQ=-DBPQ
      GO TO 470
  800 IF(IABS(LEG).EQ.2)GO TO 840
      ITAIL=TAILRS
      IHEAD=HEADRS
      STATUS=STATRS
      DELTA=DELTA1*RATIO
      FLOWRS=FLOWRS+STATRS*DELTA
      IW=IW2
      ENTER=RS
      CALL AUGMEN
      ITAIL=TAILPQ
      IHEAD=HEADPQ
      STATUS=STATPQ
      DELTA=DELTA1
      IW=IW1
      ENTER=PQ
      CMIN=CBPQ
      DMIN=DBPQ
      IF(STATUS.EQ.1)GO TO 810
      CMIN=-CMIN
      DMIN=-DMIN
  810 ICAP=IABS(CAP(PQ))
      K=K1
      RFLOW=FLOWPQ
      CALL UPDATE
      CAP(LEAVE)=-IABS(CAP(LEAVE))
      IF(TOTAL.LE.S)CAP(LEAVE)=-CAP(LEAVE)
      FLOWPQ=FLOWRS
      PQ=RS
      TAILPQ=TAILRS
      HEADPQ=HEADRS
      CBPQ=COST(PQ)+PI(TAILPQ)-PI(HEADPQ)
      DBPQ=BUD(PQ)+PID(TAILPQ)-PID(HEADPQ)
      IF(STATRS.EQ.1)GO TO 820
      CBPQ=-CBPQ
      DBPQ=-DBPQ
  820 STATPQ=-STATRS
      CBPQ=-CBPQ
      DBPQ=-DBPQ
  830 GO TO 790
  840 ENTER=PQ
      ITAIL=TAILPQ
      IHEAD=HEADPQ
```

```
      DELTA=DELTA1
      IN=IN1
      FLOWPQ=FLOWPQ+STATPQ*DELTA1
      CALL AUGMEN
      ITAIL=TAILRS
      IHEAD=HEADRS
      STATUS=STATRS
      DELTA=DELTA1*RATIO
      IN=IN2
      LEG=LEG/2
      ENTER=RS
      K=K1
      CMIN=CBRS
      DMIN=DBRS
      IF(STATUS.EQ.1)GO TO 850
      CMIN=-CMIN
      DMIN=-DMIN
  850 ICAP=IABS(CAP(RS))
      RFLOW=FLOWRS
      CALL UPDATE
      CAP(LEAVE)=-IABS(CAP(LEAVE))
      IF(TOTAL.LE.S)CAP(LEAVE)=-CAP(LEAVE)
      CBPQ=COST(PQ)+PI(TAILPQ)-PI(HEADPQ)
      DBPQ=BUD(PQ)+PID(TAILPQ)-PID(HEADPQ)
      IF(STATPQ.EQ.1)GO TO 860
      CBPQ=-CBPQ
      DBPQ=-DBPQ
  860 STATPQ=-STATPQ
      CBPQ=-CBPQ
      DBPQ=-DBPQ
  870 GO TO 770
  880 CALL RTIME(TIME2)
      TIME=TIME2-TIME1
      WRITE(22,890),C,D,TIME,ITER
  890 FORMAT(1X,'COST=',F14.4,'BUDGET=',F14.4,'TIME=',I10,'ITER=',I8)
      STOP
      END
```

```
* THIS SUBROUTINE UPDATES THE OBJECTIVE FUNCTION OF INTERAVAL         *
* GOAL PROGRAMMING                                                     *
************************************************************************
      SUBROUTINE UPDAT1(C,C1,D,D1,MINIM,W1,W2,W3,W4,OVER,DIR,DELTA,
     1         RFLAG,C2,D2)
      REAL MIN,MINIM
      INTEGER OVER
      IF(DELTA.EQ.0)GO TO 50
      IF(C.GT.C2.OR.C.LT.C1.OR.D.LT.D1.OR.D.GT.D2)GO TO 10
      OVER=1
      GO TO 50
10    IF(C.GT.C1.AND.C.LT.C2)GO TO 50
      IF(D.GT.D1.AND.D.LT.D2)GO TO 50
      IF(RFLAG)70,20,30
20    IF(DIR)30,50,70
30    IF(C.GE.C2)GO TO 60
      MIN=W1*(C1-C)+W4*(D-D2)
40    OVER=1
      IF(MIN.GE.MINIM)GO TO 50
      MINIM=MIN
      OVER=0
C     TYPE*,C,D,C1,D1,C2,D2,MIN,MINIM,OVER
50    RETURN
60    MIN=W3*(C-C2)+W4*(D-D2)
      GO TO 40
70    IF(C.GE.C2)GO TO 80
      MIN=W1*(C1-C)+W2*(D1-D)
      GO TO 40
80    MIN=W3*(C-C2)+W2*(D1-D)
      GO TO 40
      END
```

```
*********************************************************************
* THIS SUBROUTINE UPDATES THE DUAL VARIABLES PI'S AND PID'S ,      *
* THREAD INDICES, NO. OF SUCCESSORS, PREDECESSOR INDICES AND       *
* FLOW                                                              *
*********************************************************************
      SUBROUTINE UPDATE
      INTEGER WIDTH,SEED,ROOT,SOURCE,SINK,PREV,COUNT,FRONT,START,
     1        FINISH,ENTER,V,STATUS,Q,QP,Z,X,R,
     2        TIME1,TIME2,TIME,CMIN,TR,
     3        PRED(405),THREAD(405),PI(405),POINT(405),PID(0:405),
     4        SCAN(405),HEAD(3000),CAP(3000),CUST(3000),BUD(0:3000),
     5        PQ,RS,TAILPQ,TAILRS,HEADPQ,HEADRS,CODE,CODE1,CODE2
      REAL MAX,MAX1,MAX2,MUE,MUEMAX,ICAP,IFLOW,JFLOW
      DIMENSION NUMBER(405),LINK(405),LIST(405),FLOW(405),SUM(405)
      COMMON ITAIL,IHEAD,STATUS,DELTA,ICAP,LEG,ENTER,CMIN,DMIN,IW,
     1        LINK,FLOW,PRED,SUM,CAP,NUMBER,THREAD,PI,PID,IFLOW,K
     2        ,LEAVE,RFLOW
      IFLOW=RFLOW+STATUS*DELTA
C     IF(STATUS.EQ.-1)IFLOW=ICAP-DELTA
      ILINK=-STATUS*LEG*ENTER
      IF(ILINK.GT.0)GO TO 610
      Q=ITAIL
      QP=IHEAD
      CMIN=-CMIN
      DMIN=-DMIN
      GO TO 620
610   Q=IHEAD
      QP=ITAIL
620   CHANGE=LEG*DELTA
      KP=PRED(K)
      LEAVE=IABS(LINK(K))
      KODE=LEG
      IF(LINK(K).LT.0)KODE=-KODE
      IF(KODE.EQ.1)CAP(LEAVE)=-CAP(LEAVE)
      CAP(ENTER)=ICAP
      IPRED=QP
      INUMB=NUMBER(K)
      L=0
      I=Q
625   JFLOW=FLOW(I)
      JLINK=LINK(I)
      JPRED=PRED(I)
      JSUCC=NUMBER(I)-L
      Z=I
      X=THREAD(I)
      COUNT=2
630   IF(COUNT.GT.JSUCC)GO TO 640
      PI(X)=PI(X)+CMIN
      PID(X)=PID(X)+DMIN
      Z=X
      X=THREAD(X)
      COUNT=COUNT+1
      GO TO 630
640   R=JPRED
650   TR=THREAD(R)
      IF(TR.EQ.I)GO TO 660
      R=TR
      GO TO 650
660   FLOW(I)=IFLOW
      LINK(I)=ILINK
      NUMBER(I)=INUMB
      PRED(I)=IPRED
      PI(I)=PI(I)+CMIN
      PID(I)=PID(I)+DMIN
      THREAD(R)=X
      IF(I.EQ.K)GO TO 690
      THREAD(Z)=JPRED
      IF(JLINK.GT.0)GO TO 670
      IFLOW=JFLOW+CHANGE
      GO TO 680
670   IFLOW=JFLOW-CHANGE
680   ILINK=-JLINK
      INUMB=INUMB-JSUCC
      L=L+JSUCC
      IPRED=I
      I=JPRED
      GO TO 625
690   THREAD(Z)=THREAD(QP)
      THREAD(QP)=Q
      CHANGE=-CHANGE
      NUMBQ=NUMBER(Q)
      I=QP
710   IF(I.EQ.IN)GO TO 740
      NN=NN+1
      IF(LINK(I).GT.0)GO TO 720
      FLOW(I)=FLOW(I)+CHANGE
      NUMBER(I)=NUMBER(I)+NUMBQ
      I=PRED(I)
```

```
  720 FLOW(I)=FLOW(I)-CHANGE
      NUMBER(I)=NUMBER(I)+NUMBQ
      I=PRED(I)
      GO TO 710
  740 I=KP
      CHANGE=-CHANGE
  750 IF(I.EQ.IW)RETURN
      IF(LINK(I).LT.0)GO TO 760
      FLOW(I)=FLOW(I)+CHANGE
      NUMBER(I)=NUMBER(I)-NUMBQ
      I=PRED(I)
      GO TO 750
  760 FLOW(I)=FLOW(I)-CHANGE
      NUMBER(I)=NUMBER(I)-NUMBQ
      I=PRED(I)
      GO TO 750
      END
```

```
*******************************************************************
* THIS SUBROUTINE UPDATES THE FLOW                                *
*******************************************************************
      SUBROUTINE AUGMEN
      INTEGER WIDTH,SEED,ROOT,SOURCE,SINK,PREV,COUNT,FRONT,START,
     1         FINISH,ENTER,V,STATUS,Q,QP,Z,X,R,
     2         TIME1,TIME2,TIME,CMIN,TR,
     3         PRED(405),THREAD(405),PI(405),POINT(405),PID(0:405),
     4         SCAN(405),HEAD(3000),CAP(3000),COST(3000),BUD(0:3000),
     5         PQ,RS,TAILPQ,TAILRS,HEADPQ,HEADRS,CODE,CODE1,CODE2
      REAL MAX,MAX1,MAX2,MUE,MUEMAX,ICAP,IFLOW1,JFLOW
      DIMENSION NUMBER(405),LINK(405),LIST(405),FLOW(405),SUM(405)
      COMMON ITAIL,IHEAD,STATUS,DELTA,ICAP,LEG,ENTER,CMIN,DMIN,IW,
     1         LINK,FLOW,PRED,SUM,CAP,NUMBER,THREAD,PI,PID,IFLOW,K
     2         ,LEAVE,RFLOW
      CAP(ENTER)=-CAP(ENTER)
      CHANGE=DELTA*STATUS
      I=ITAIL
410   IF(I.EQ.IW)GO TO 440
      J=LINK(I)
      IF(J.LT.0)GO TO 420
      FLOW(I)=FLOW(I)+CHANGE
      I=PRED(I)
      GO TO 410
420   FLOW(I)=FLOW(I)-CHANGE
      I=PRED(I)
      GO TO 410
440   I=IHEAD
450   IF(I.EQ.IW)RETURN
      J=LINK(I)
      IF(J.LT.0)GO TO 460
      FLOW(I)=FLOW(I)-CHANGE
      I=PRED(I)
      GO TO 450
460   FLOW(I)=FLOW(I)+CHANGE
      I=PRED(I)
      GO TO 450
      END
```

```
***********************************************************************
* THIS PART OF THE PROGRAM ( SUBROUTINES MG4 , STAR AND               *
* FUNCTION IRAN ) GENERATES THE NETWORK.                              *
***********************************************************************
      SUBROUTINE MG4(WIDTH,LENGTH,N,M,S,T,POINT,HEAD)
***********************************************************************
* THIS SUBROUTINE GENERATES THE HEAD NODES OF ARCS                    *
***********************************************************************
      INTEGER S,T,COUNT,WIDTH,
     1        ALIST(3000),BLIST(3000),POINT(405),RPOINT(405),
     2        TAIL(3000),HEAD(3000),CAP(3000),TRACE(3000)
      LARGE=1000000
      N=LENGTH*WIDTH+2
      S=1
      T=N
      IA=LARGE
      IB=LARGE
      IC=0
      ID=100
      IW=2
      IX=WIDTH+1
      IY=N-WIDTH
      IZ=N-1
      DO 10 I=1,N
      ALIST(I)=0
   10 CONTINUE
      COUNT=WIDTH*(LENGTH+1)+1
      DO 20 J=COUNT,M
      ITAIL=IRAN(1,N-1)
      ALIST(ITAIL)=ALIST(ITAIL)+1
   20 CONTINUE
      COUNT=0
      DO 30 I=IW,IX-1
      COUNT=COUNT+1
      TAIL(COUNT)=1
      HEAD(COUNT)=I
      CAP(COUNT)=IRAN(IA,IB)
   30 CONTINUE
      DO 150 K=1,IZ
      DO 50 I=1,N
   50 BLIST(I)=0
      I1=MIN0(K+WIDTH,N)
      BLIST(I1)=1
      IF(ALIST(K).EQ.0)GO TO 110
      DO 100 J=1,ALIST(K)
   90 IHEAD=IRAN(2,N)
      IF(BLIST(IHEAD).EQ.1)GO TO 90
      IF(IHEAD.EQ.K)GO TO 90
      BLIST(IHEAD)=1
  100 CONTINUE
  110 DO 120 I=N,2,-1
      IF(BLIST(I).EQ.0)GO TO 120
      COUNT=COUNT+1
      TAIL(COUNT)=K
      HEAD(COUNT)=I
      IF(TAIL(COUNT).EQ.1.AND.HEAD(COUNT).EQ.IX)GO TO 130
      IF(HEAD(COUNT).EQ.N.AND.HEAD(COUNT)-TAIL(COUNT).LE.WIDTH)
     1   GO TO 130
      CAP(COUNT)=IRAN(IC,ID)
      GO TO 120
  130 CAP(COUNT)=IRAN(IA,IB)
  120 CONTINUE
  150 CONTINUE
      CALL STAR(N,M,TAIL,HEAD,CAP,TRACE,POINT,RPOINT)
      RETURN
      END
```

```
      ******************************************************************
      * THIS SUBROUTINE GENERATES POINTERS OF THE NETWORK              *
      ******************************************************************
            SUBROUTINE STAR(N,M,TAIL,HEAD,CAP,TRACE,POINT,RPOINT)
            INTEGER COUNT,POINT(405),RPOINT(405),ALIST(3000),BLIST(3000),
           1        TAIL(3000),HEAD(3000),TRACE(3000),CAP(3000)

            DO 10 I=1,N
            ALIST(I)=0
      10    CONTINUE
            POINT(1)=1
            K=1
            DO 20 J=1,M
            ALIST(HEAD(J))=ALIST(HEAD(J))+1
            IF(TAIL(J).EQ.K)GO TO 20
            K=K+1
            POINT(K)=J
      20    CONTINUE
            POINT(N)=M+1
            POINT(N+1)=M+1
      C     CALL SORT(N,M,POINT,TAIL,HEAD,CAP)
            BLIST(1)=1
            RPOINT(1)=1
            DO 30 I=2,N
            BLIST(I)=BLIST(I-1)+ALIST(I-1)
            RPOINT(I)=BLIST(I)
      30    CONTINUE
            RPOINT(N+1)=M+1
            DO 40 J=M,1,-1
            JHEAD=HEAD(J)
            TRACE(BLIST(JHEAD))=J
            BLIST(JHEAD)=BLIST(JHEAD)+1
      40    CONTINUE
            RETURN
            END
```

```
****************************************************************************
* THIS FUNCTION GENERATES A RANDOM NUMBER BETWEEN IC AND ID.               *
****************************************************************************

      FUNCTION IRAN(IC,ID)
      IRAN=IC+(ID-IC+1)*RAN(X)
      IF(IRAN.GT.ID)IRAN=ID
      RETURN
      END
```